Log-Distributional Approach for Learning Covariate Shift Ratios

Author: Guillermo Bernárdez Gil

1st **Advisor: Carles Gelada**
Google Brain, Montreal

2nd **Advisor: Sergio Escalera**
Departament de Matemàtiques i Informàtica
Universitat de Barcelona

A thesis presented for the degree of
Master in Artificial Intelligence

Facultat d'Informàtica de Barcelona (FIB)
Escola Tècnica Superior d'Enginyeria (URV)
Facultat de Matemàtiques i Informàtica (UB)

October 21, 2019

Abstract

Temporal Difference (TD) Learning algorithms classically learn value functions (sum of discounted rewards) by solving for the fixed point that relates the value of one state with the expected value of the following states. Other TD learning algorithms have been used to learn *multiplicative value functions*, which also have a fixed point solution, even though it has been shown that learning of these multiplicative value functions suffers from higher variance and looser convergence guarantees than additive ones. One potential approach to solving these issues would be to learn in log space, turning the multiplicative value function into an additive one. Unfortunately, it is easy to show with Jensen's inequality that the exponential of the log fixed point does not correspond to the multiplicative fixed point.

On the other hand, Distributional Reinforcement Learning (RL) has been used to learn the return distribution (return is the sum of future discounted rewards). Although it has been shown that it mainly plays an auxiliary task role for representation learning, we propose that distributional fixed points could play a much more fundamental role to learning non additive value functions. In the case of distributional multiplicative value functions, learning the additive fixed point in log space and exponentiating leads to the correct solution.

In particular, we study this approach in the case of the learning process of the the ratio between stationary distributions of two policies (Covariate Shift Ratio), which defines a multiplicative value function. In our theoretical analysis we are able to get stronger convergence guarantees than any previous work on the Covariate Shift, and we develop a practical framework using the Arcade Learning Environment (ALE) to evaluate it. As a result, we observed good trends in the learning of the ratio, obtaining similar performance than previous state-of-the-art works in the area.

Aknowledgements

I would first like to thank my advisor Carles Gelada for all his guidance and implication, I am gratefully indebted to him for everything he has taught me along this exciting journey.

I would also like to acknowledge my advisor Sergio Escalera for his passionate participation and support, always providing very valuable advice.

Finalmente, quiero dedicar estas últimas palabras directamente a mi familia, especialmente a mis padres y abuelos: gracias por estar siempre a mi lado, gracias por apoyarme incondicionalmente en todos mis proyectos. Esto va por y para vosotros.

Contents

1 Introduction

The concept of Reinforcement Learning[1] (RL) emerged with the objective of designing agents able to deal with sequential decision problems by themselves, in the sense that they are not told which actions to take; instead, they must learn good policies by optimizing a cumulative future reward signal.

In particular, one of the main challenges that a RL must face is the trade-off between exploration and exploitation[1]: to maximize the reward, it has to exploit its previous experience while exploring for possible better action selections. The risen of this dilemma, which is not present in other kinds of learning (neither supervised nor unsupervised setting have to deal with it), has already linked the theory of RL with psychological[2] and neuroscientific[3] perspectives on animal behaviour.

In fact, the vast majority of RL algorithms and approaches, if not all, try to model how humans and other animals optimize their control of an environment; great examples of that are the widely use in RL of Temporal Difference (TD) learning[4], which is based on how living beings deal with rewards, as well as the neurological inspiration behind Deep Reinforcement Learning[5] implementations.

Despite the high number of RL real-world applications that already exist (e.g. [6, 7, 8, 9, 10]), most research of the community is still focused on getting new insights into how RL agents should be implemented to improve both their understanding of the environment and their learning capabilities[11, 12, 13, 14, 15, 16]. In particular, new contributions are commonly benchmarked using very well-defined domains -usually human games- that facilitate their comparison with other approaches.

One relevant -and relatively recent- contribution was the Distributional RL setting presented in [17], which attained state-of-the-art performance on several RL benchmarks[18, 19]. This work motivated a change of paradigm for solving RL problems: for the agent to guide its behaviour, they argued the importance of it learning the full distribution of return -meaning by return the discounted sum of rewards- instead of only its expectation -as the typical value-based RL does[1].

Even though some distributional approaches were previously proposed[20, 21], it was not until [17] was published that the study of distributional-based solutions actually became a very active focus of research within the RL community[18, 22, 23, 24], providing evidence that they positively impact the RL approximate setting. However, despite the fundamental theoretical results already shown[25, 26], there is still much work to do in understanding the ins and outs of such a distributional perspective, and we believe that its implications might go beyond the task of representation learning.

In particular, distributional-based RL introduced the notion of distributional Bellman equations[17], which are encoded in terms of mixture distributions[25]. A quick analysis of distributional learning reveals that the corresponding distributional fixed points satisfy a promising property for learning non-additive value functions.

In the context of off-policy RL learning[1], where the agent learns from data that might not be drawn from its current policy, the Covariate Shift approach presented in [27] precisely defines a TD-based, multiplicative update rule for learning the ratio between the stationary distributions generated by two policies.

It is well documented that off-policy learning suffers from divergence issues when combined with function approximation and bootstrapping (e.g. TD methods) -authors of [1] named this context as the 'deadly triad' of RL. According to [4], the main reason why convergence

guarantees break under this regime relies in the fact that usual Bellman learning rules require from updating states according to the stationary distribution of the learned policy (target policy), whereas the sampling is performed using data that follows another policy (behaviour policy).

In short, the aforementioned covariate shift method manages this problem by reweighting updates according to the ratio of the target and behavior stationary distributions, in such a way that, in expectation, we recover a sampling under the target stationary distribution[27]. In practice, this reweighting can be easily implemented through a prioritized experience replay scheme[28] where priorities are estimated ratios.

The original Covariate Shift approach, though, only showed weak convergence guarantees, and was difficult to implement together with deep reinforcement learning. These issues were addressed in [29], where both the stability and the applicability of the method were improved by slightly modifying some minor aspects of the algorithm.

Nevertheless, even with the improved version of [29], the resulting methodology still lacks from the desired convergence guarantees, and the updates suffers from a high variance. The main reason of this happening might relate with the fact that the learning rule is multiplicative; in fact, if rewriting it as an additive value function were possible, we strongly believe that a better learning behaviour would be obtained.

Motivation and Goals

The connection between TD-based distributional learning rules and mixtures distributions, made in [25] for the case of distributional-based RL, motivated us to further investigate what implications it might have. Surprisingly, one can easily infer, through mixture properties, that distributional learning allows to project the learning rule into another space (by applying a continuous, invertible function), find the fixed point in that space, and get the correct solution by simply reverting the projection of that fixed point; Jensen's inequalities, in contrast, prevents from a similar result in value-based learning processes. This property, as mentioned before, is specially appealing when dealing with non-additive value functions.

In particular, we are interested in defining a theoretically-grounded distributional approach for learning covariate shift ratios, showing how we can turn the multiplicative behaviour of their original update rule into an additive one by simply moving to the log space. Besides studying the theoretical convergence guarantees of this precise distributional approach, we are also concerned about providing evidence of its practicability, so another goal is to develop an algorithmic implementation of the method as well as design an experimental set up for its evaluation.

Structure of the Report

In Section 2 we provide a thorough review of Distributional Reinforcement Learning, detailing the fundamental theoretical results on this area. Section 3 is devoted to introduce the covariate shift approach, paying special attention to its convergence guarantees with linear function approximation. In section 4 we present our main contribution: a theoretical-grounded, distributional framework for learning covariate shift ratios in the logarithmic space (i.e. with an additive update rule). Section 5 describes the details of our implementation, which we assess in Section 6 by showing and analyzing the results of the different experiments performed. Finally, Section 7 summarizes the conclusions as well as the possible next steps to take.

2 Distributional Reinforcement Learning

The aim of this section is to provide a detailed review of the main concepts and theoretical results of Distributional Reinforcement Learning, focusing on those approaches where learned distributions are approximated within a categorical parametric family -just as the first renowned distributional-based RL algorithm (C51) presented in [17].

Hence, here we revisit some relevant papers on this area, collecting and explaining their main contributions. Among all consulted references, we make special allusion to the works of [17], responsible of making distributional RL so popular by introducing C51 to the community; [25], a thorough theoretical analysis of categorical distributional RL algorithms; and [26], which presents the first converge result of a categorical distributional-based approach together linear function approximation.

The order of the section is as follows: first, we introduce the general Reinforcement Learning setting. Second, we describe how we go beyond the expected value-based RL to its distributional perspective. Third, we present a characterization of distributional RL algorithms according to the approximations that are taken. Finally, we present the main convergence guarantees of Categorical Distributional RL that have been found so far by the community, both in the tabular case and with linear function approximation.

Remark. It is worth noting that having clear the concepts of contraction mappings and mixture distributions are key for fully understanding some important aspects of the theory presented both in this section and in the following ones. For that reason, we encourage readers that are not familiarized with these notions to visit the Appendix; in particular, we characterize contraction mappings in Appendix A.1, describe mixture distributions and their properties in Appendix A.2, and also provide the definition of some relevant metrics over distributions in Appendix A.3.

2.1 Reinforcement Learning Setting

Let us begin by characterizing the RL framework.

We consider an agent interacting with an environment in the standard setting[4]: at each step t, the agent selects an action a_t based on its current state s_t, to which the environment responds with a reward r_t and then moves to the next state s_{t+1}. This interaction is modeled as a time-homogeneous Markov Decision Process $(\mathcal{S}, \mathcal{A}, r, P, \gamma)$, where

- $\mathcal{S}$ and $\mathcal{A}$ are the state and action spaces, respectively. It is assumed that both are finite, with $n := |\mathcal{S}|$;
- P is the transition kernel, $s_{t+1} \sim P(\cdot|s_t, a_t)$; the Markov assumption states that $P(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, ...) = P(s_{t+1}|s_t, a_t)$;
- r_t represents the immediate reward given by the environment after taking action a_t being in state s_t. These rewards are considered to be sampled from a reward distribution $R(s, a)$, i.e. $r_t \sim R(s_t, a_t)$;
- γ is the discount factor.

A policy π maps each state to a probability distribution over the action space, $a_t \sim \pi(\cdot|s_t)$. In particular, a stationary policy[1] π defines in turn a Markov Reward Process (MRP), that

[1]By stationary we simply mean that we consider the same policy for all time-steps.

is, a Markov Chain with reward associated with every transition. The transition function of this Markov chain is the so-called state-to-state transition matrix $P^\pi \in \mathbb{R}^{n\times n}$, whose entries are

$$P^\pi_{s,s'} := P_\pi(s'|s) = Prob_\pi(s_{t+1} = s'|s_t = s) = \sum_{a\in\mathcal{A}} \pi(a|s)P(s'|s,a),$$

and whose n-th power, $(P_\pi)^n$, encodes the transition function across n time-steps. Regarding the reward of the MRP, it is given by the expected immediate reward vector $r^\pi \in \mathbb{R}^n$, with

$$r^\pi_s := r^\pi(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}\left[R(s,a)\right]$$

A stationary distribution of a Markov Chain with state space $\mathcal{S}$ and transition matrix P, if exists, is defined as some vector $d \in \Delta(\mathcal{S})$, where $\Delta(\mathcal{S}) \subset \mathbb{R}^n$ accounts for the simplex over states[2], that satisfies $d = P^T d$; generally speaking, stationary distributions describe the proportion of time that the chain spends in each state $s \in \mathcal{S}$ over a long run. In our particular case, under ergodicity and some other mild conditions[30], we can guarantee that a stationary policy π induces a unique stationary distribution of its associated MRP after enough time-steps (i.e. in the limit $t \to \infty$); it is referred to as the stationary distribution of the policy π. and denoted by d_π.

Back to the general MDP, another relevant concept is that of the return, usually defined as simply the sum of all received rewards along a certain agent trajectory $\{(s_t, a_t, r_t)\}_{t=0}^T$. In practice, however, it is common to consider its discounted counterpart, which uses the discount factor γ to give higher weight to near rewards than those received further in the future:

$$\sum_{t=0}^{T} \gamma^t r_t$$

It is worth noting that the main reason to contemplate discounted returns with $\gamma < 1$ is purely mathematical: this way infinite reward sums (when $T \to \infty$) become finite, which helps proving the convergence of certain RL algorithms.

The estimation of an agent's discounted return lead us to the ubiquitous notions of both action and state value functions. On the one hand, the action value function is defined as the expected discounted return from a state-action pair by following a certain policy π, i.e.

$$Q^\pi(s,a) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t R(s_t,a_t)\middle| s_0 = s, a_0 = a\right] \tag{2.1}$$

From previous expression 2.1, a Bellman's equation for the action value function can be easily derived:

$$\begin{aligned}
Q^\pi(s,a) &= \mathbb{E}\left[R(s,a)\right] + \gamma \mathop{\mathbb{E}}_{\pi}\left[\sum_{t=1}^{\infty} \gamma^t R(s_{t+1},a_{t+1})\middle| s_0 = s, a_0 = a\right] \\
&= \mathbb{E}\left[R(s,a)\right] + \gamma \sum_{s'} P(s'|s,a)\left(\mathop{\mathbb{E}}_{\pi}\left[\sum_{t=1}^{\infty} \gamma^t R(s_{t+1},a_{t+1})\middle| s_1 = s'\right]\right) \\
&= \mathbb{E}\left[R(s,a)\right] + \\
&\quad \gamma \sum_{s'} P(s'|s,a) \sum_{a'} \pi(a'|s')\left(\mathop{\mathbb{E}}_{\pi}\left[\sum_{t=1}^{\infty} \gamma^t R(s_{t+1},a_{t+1})\middle| s_1 = s', a_1 = a'\right]\right) \\
&= \mathbb{E}\left[R(s,a)\right] + \gamma \mathop{\mathbb{E}}_{\substack{s'\sim P(\cdot|s,a)\\ a'\sim\pi(\cdot|s')}}\left[Q(s',a')\right]
\end{aligned} \tag{2.2}$$

[2] $d \in \Delta(\mathcal{S}) \implies d^T e = 1, d \geq 0$, denoting by $e \in \mathbb{R}^n$ the vector of all ones.

Moreover, among all action value functions, there exists at least one optimal action value function Q^* in the sense that it has the higher possible value for all states:

$$Q^*(s,a) := \max_\pi Q^\pi(s,a).$$

On the other hand, the state value function is defined as the expected sum of discounted rewards from a state by following π, and also gives rise to a Bellman's equation:

$$\begin{aligned} V^\pi(s) &= \mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t R(s_t,a_t)\middle| s_0=s\right] \\ &= \mathop{\mathbb{E}}_{a\sim\pi(\cdot|s)}[R(s,a)] + \\ &\quad \gamma\sum_a \pi(a|s)\sum_{s'} P(s'|s,a)\left(\mathop{\mathbb{E}}_\pi\left[\sum_{t=1}^{\infty}\gamma^t R(s_{t+1},a_{t+1})\middle| s_1=s'\right]\right) \\ &= r^\pi(s) + \gamma \mathop{\mathbb{E}}_{s'\sim P_\pi(\cdot|s)}\left[V^\pi(s')\right] \end{aligned} \tag{2.3}$$

Analogously, there also exists an optimal state value function V^* so that, for all states,

$$V^*(s) := \max_\pi V^\pi(s).$$

In vector notation, we have $Q^\pi \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ and $V^\pi \in \mathbb{R}^{\mathcal{S}}$, so the usual Euclidean L_2 norm could be used for comparing two value functions. In practice, however, one might be interested in giving more importance to some states than others -for instance. because they occur more frequently. For that reason, as discussed in [1], it is common to contemplate a distribution $d \in \Delta(\mathcal{S})$ (usually the stationary distribution d_π) to specify the degree to which we care about different states being accurately valued, and then define the distance between any pair of value functions using the d-weighted L_2 norm[3]; e.g.

$$\|V - V'\|_d^2 = \sum_{s\in\mathcal{S}} d(s)(V(s) - V'(s))^2$$

for any $V, V' \in \mathbb{R}^{\mathcal{S}}$.

We emphasize that the objective of RL is to find a policy that, given any considered state $s \in \mathcal{S}$, always select an action that maximize the expected (discounted) return; such a policy is called an optimal policy, an we denote all of them (there might be more than one) by π^*. In particular, note that finding the optimal policy is equivalent to finding the policy with an optimal value function, in the sense that for all states $s \in \mathcal{S}$

$$\pi^* = \max_\pi V^\pi(s), \quad \text{Prob}(\arg\max_a Q^*(s,a) \sim \pi^*(\cdot|s)) = 1$$

In fact, most RL algorithms actually make use of value functions to estimate the optimal policy, and they do so by applying operators based on Bellman's equations 2.2 and 2.3. As we will see with the action value Bellman equation case, they play a fundamental role in several key aspects of the learning process.

[3]In general, given an arbitrary $d \in \mathbb{R}^k$, the d-weighted L_2 norm of any vector $x \in \mathbb{R}^k$ is defined as $\|x\|_d^2 := \sum_{i=1}^k d(i)x(i)^2$.

Bellman Operators

Typically, one can distinguish two different settings among -and within- RL algorithms. On the one hand, we may be interested in computing the value function, assuming the current policy π is fixed, so as to evaluate that policy (*policy evaluation*). On the other hand, we might want to actually improve the current policy (*control setting*). Both settings have been extensively analyzed (e.g. [1]) and have associated their corresponding Bellman operators.

Remark. We present and briefly analyze Bellman operators in the case of action value functions, but the shown development is analogous for state value functions V^π.

In the context of *policy evaluation*, considering state-action value functions Q^π as vectors in $\mathbb{R}^{\mathcal{S}\times\mathcal{A}}$, the *Bellman operator* $\mathcal{T}^\pi : \mathbb{R}^{\mathcal{S}\times\mathcal{A}} \to \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ is defined as

$$\mathcal{T}^\pi Q(s,a) := \mathbb{E}\left[R(s,a)\right] + \gamma \mathop{\mathbb{E}}_{\substack{s'\sim P(\cdot|s,a)\\ a'\sim\pi(\cdot|s')}} \Big[Q(s',a')\Big] \tag{2.4}$$

Such operator is useful to describe the expected behaviour of popular learning algorithms (e.g. SARSA), and as stated in [4] satisfies a very interesting property: $\mathcal{T}^\pi$ is a contraction mapping (see Appendix A.1) on the d_π-weighted Euclidean distance, i.e.

$$\|\mathcal{T}^\pi Q - \mathcal{T}^\pi Q'\|_{d_\pi} \leq \gamma \|Q - Q'\|_{d_\pi} \text{ for any } Q, Q' \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}.$$

In particular, this fact allows to demonstrate, by simply applying Banach's Fixed-point Theorem, that the process $Q_t = \mathcal{T}^\pi Q_{t-1}$, for some initial value Q_0, converges exponentially to Q^π as $t \to \infty$.

Regarding the *control setting*, where the goal is to improve the current policy π, a *Bellman optimality operator* $\mathcal{T}^* : \mathbb{R}^{\mathcal{S}\times\mathcal{A}} \to \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ is defined as follows:

$$\mathcal{T}^* Q(s,a) := \mathbb{E}\left[R(s,a)\right] + \gamma \mathop{\mathbb{E}}_{s'\sim P(\cdot|s,a)} \Big[\max_{a'\in\mathcal{A}} Q(s',a')\Big] \tag{2.5}$$

This operator, for instance, describes the expected behaviour of Q-learning. Moreover, as it was shown in [4], $\mathcal{T}^*$ is also a contraction mapping on the d_π-weighted Euclidean distance, and hence the sequence $\{Q_t\}$, where $Q_t = \mathcal{T}^* Q_{t-1}$, converges exponentially to a fixed point, which is precisely the optimal value function Q^*[4].

However, in spite of the provided depiction of both settings, we emphasize that *policy evaluation* is also used in most policy improvement algorithms. This is the main reason why, along the following theoretical development, we will be more focused on the study of Bellman operators than on their optimality counterparts. By doing that, we simplify our dissertation and avoid the repetition of some tedious results, but we also expect that strong convergence guarantees of T^π might correlate with a good-bahaved T^*.

Function Approximation

In most RL problems, the large size of their state spaces makes unfeasible the so-called tabular case, where we can learn separately either the value V^π of each state or the value Q^π of each state-action pair, depending on the value function considered. In these cases, we need to get value function estimates through *function approximators*, and among them it is common to consider a *linear function approximation*.

Remark. We will use state value function V^π to explain some basic aspects of RL with linear function approximation since its related notation is slightly simpler than that of Q^π. However, the same exact comments and conclusions apply for action value functions.

In the case of state value functions, a typical linear function approximation uses a linear mapping from states to features $\phi : \mathcal{S} \to \mathbb{R}^k$. In these cases, the approximate action value function at a state s can be expressed as the inner product of a feature vector with a vector of weights $\theta \in \mathbb{R}^k$:

$$\hat{V}(s) = \phi(s)^T\theta \tag{2.6}$$

which can be written as $\hat{V} = \Phi\theta$ in vector notation, being $\Phi \in \mathbb{R}^{n\times k}$ the matrix with $\theta(s)$ as row vectors. Under this approximation regime, the *semi-gradient update rule* for TD learning[1] learns an estimation of V^π stochastically from sample transitions; given a starting state $s \in \mathcal{S}$, a successor state $s' \sim P_\pi(\cdot|s)$, and a step-size parameter $\alpha > 0$, this update is defined as

$$\theta \leftarrow \theta + \alpha \left[r^\pi(s) + \gamma\phi(s')^T\theta - \phi(s)^T\theta\right] \phi(s) \tag{2.7}$$

This gradient descent-based rule is precisely designed to deal with the fact that most value functions might not be representable by the considered function approximation. In fact, under the described linear approximation regime, there is the extra task of, given any non-representable value function V, finding its closest representation in the subspace $\mathcal{W}_\Phi := \{V \in \mathbb{R}^n : V = \Phi\theta\}$ of representable value functions. As we will see, this can be modeled by a projection step.

We recall that the distance between two value distributions $V, V' \in \mathbb{R}^n$ is typically defined as the d-weighted Euclidean distance for some $d \in \Delta(\mathcal{S})$. Then, it can be defined a projector operator $\Pi_d : \mathbb{R}^n \to \mathcal{W}_\Phi$ so that, given any value function V, it provides with its closest representable function according to the d-weighted norm:

$$\Pi_d V := \arg\min_{\hat{V}\in\mathcal{W}_\Phi} \|V - \hat{V}\|_d^2.$$

Each time that the update rule 2.7 is applied, it performs a step towards the minimization that represents the projection operator Π_d.

Therefore, the expected behaviour of 2.7 is actually described by the *projected Bellman operator* $\Pi_d\mathcal{T}_\pi$, a combination of the usual Bellman operator with the projection Π_d onto the span of Φ[31]. In fact, any convergence analysis requires now the consideration of the *projected Bellman equation* $\hat{V}^\pi = \Pi_d\mathcal{T}_\pi\hat{V}^\pi$.

We emphasize that convergence of $\Pi_d\mathcal{T}^\pi$ is only guaranteed when the projection weights are $d = d_\pi$. As usual, the proof is done by first showing that the combined operator $\Pi_{d_\pi}T^\pi$ is a contraction in the d_π-weighted Euclidean norm, i.e. for any two value functions $V, V' \in \mathbb{R}^n$

$$\|\Pi_{d_\pi}T^\pi V - \Pi_{d_\pi}T^\pi V'\|_{d_\pi} \leq \gamma\|V - V'\|_{d_\pi}$$

and then applying the Banach's fixed-point theorem to the sequence $\hat{V}_t := \Pi_{d_\pi}T^\pi\hat{V}_{t-1}$ (with some initial $\hat{V}_0 \in \mathcal{W}_\Phi$).

Remark. As it is shown in [31], for a combined operator to be a contraction in a certain metric, each of the operators involved must be a contraction in that same metric. In our case, the fact that T^π is only a contraction in the d_π-weighted L_2 norm clearly conditions the choice of Π_{d_π}, as well as the metric used to prove the contractibility of $\Pi_d\mathcal{T}^\pi$.

However, it is important to note that, even with convergence guarantees, the true value function V^π might not lie in the representable subspace $\mathcal{W}_\Phi$, so our estimate $\hat{V}^\pi$ may always differ from V^π to some extent.

2.2 Towards a Distributional Reinforcement Learning

Now that we have introduced the standard expected-based RL setting, we go a step further and present its distributional counterpart, which has become the focus of many RL research since the publication of [17] -some examples are [25, 26], but the list of related papers is much larger.

This represents a change of paradigm: Distributional RL cares about the whole distribution of returns -instead of just their expected values- to find the optimal policy. Hence, the first key point of [17] is to redefine the RL framework from a distributional perspective, i.e. so as to be able to deal with proper distributions.

Notation. Given a random variable U, we denote its probability density function (pdf) as f_U. Moreover, we indicate by a distributional equation $U \overset{D}{=} V$ that U is distributed according to the same law as another random variable V; in particular, this implies $f_U = f_V$.

However, authors of the cited work interestingly show that distributions of returns also satisfy a Bellman equation, and therefore can be computed in a very similar manner than Q-values. In particular, they define the following distributional Bellman equation

$$Z^\pi(s,a) \overset{D}{=} R(s,a) + \gamma Z^\pi(S',A'), \tag{2.8}$$

where

- $Z^\pi : \mathcal{S} \times \mathcal{A} \to \mathcal{Z}$ is the *value distribution*, a mapping from state-action pairs to distributions over returns by following policy π. As stated in [17], its expectation is the value Q^π
$$Q^\pi(s,a) := \mathbb{E}[Z^\pi(s,a)] \tag{2.9}$$
We will also call it the *return distribution*.
- $(S',A') \in \mathcal{S} \times \mathcal{A}$ is the next state-action random variable: $S' \sim P(\cdot|s,a)$, $A' \sim \pi(\cdot|S')$
- $R(s,a) \in \mathcal{Z}$ is the random reward, or equivalently the reward function. Note that now we are dealing with it as an explicit random variable.
- $Z^\pi(S',A') \in \mathcal{Z}$ is the random return over the random next state-action following π. This notation implies that all possible next state-action pairs need to be considered as to generate this return distribution. Thus, $Z^\pi(S',A')$ may be seen as a mixture distribution (see Appendix A.2) of the distributions $Z^\pi(s',a')$, where s' and a' are sampled from (S',A'):
$$f_{Z^\pi(S',A')}(z) = \sum_{s'} P(s'|s,a) \sum_{a'} \pi(a'|s') f_{Z^\pi(s',a')}(z) \tag{2.10}$$
Its expected value, using 2.9, can be then expressed as:
$$\begin{aligned}
\mathbb{E}[Z^\pi(S',A')] &= \int_{-\infty}^{\infty} z \sum_{s'} P(s'|s,a) \sum_{a'} \pi(a'|s') f_{Z^\pi(s',a')}(z) dz \\
&= \sum_{s'} P(s'|s,a) \sum_{a'} \pi(a'|s') \int_{-\infty}^{\infty} z f_{Z^\pi(s',a')}(z) dz \\
&= \sum_{s'} P(s'|s,a) \sum_{a'} \pi(a'|s') \, \mathbb{E}[Z^\pi(s',a')] \\
&= \mathop{\mathbb{E}}_{s' \sim P(\cdot|s,a), a' \sim \pi(\cdot|s')} \Big[Q^\pi(s',a') \Big]
\end{aligned} \tag{2.11}$$

Note that the classical Bellman's equation 2.2 for the action value Q can be easily recovered by using 2.9 and 2.11 when taking the expected value over its distributional version 2.8:

$$\begin{aligned} Q^\pi(s,a) &= \mathbb{E}[Z^\pi(s,a)] \\ &= \mathbb{E}[R(s,a)] + \gamma\, \mathbb{E}[Z^\pi(S',A')] \\ &= \mathbb{E}[R(s,a)] + \gamma \mathop{\mathbb{E}}_{s'\sim P(\cdot|s,a), a'\sim\pi(\cdot|s')} \Big[Q^\pi(s',a') \Big] \end{aligned} \tag{2.12}$$

Finally, looking for some clues of what actually the random return Z represents, we expand its density function as follows:

$$\begin{aligned} f_{Z^\pi(s,a)}(z) &= f_{R(s,a)+\gamma Z^\pi(S',A')}(z) \\ &= \sum_{s'} P(s'|s,a) \sum_{a'} \pi(a'|s') f_{R(s,a)+\gamma Z^\pi(s',a')}(z) \\ &= \sum_{s'} P(s'|s,a) \sum_{a'} \pi(a'|s') f_{R(s,a)+\gamma(R(s',a')+\gamma Z^\pi(S'',A''))}(z) \\ &= \sum_{s'} P(s'|s,a) \sum_{a'} \pi(a'|s') \sum_{s''} P(s''|s',a') \sum_{a''} \pi(a''|s'') \\ &\quad f_{R(s,a)+\gamma R(s',a')+\gamma^2 Z^\pi(s'',a'')}(z) \\ &= \sum_{s_1} P(s_1|s_0,a_0) \sum_{a_1} \pi(a_1|s_1) \cdots \sum_{s_t} P(s_t|s_{t-1},a_{t-1}) \sum_{a_t} \pi(a_t|s_t) \\ &\quad f_{\sum_{i=0}^t \gamma^t R(s_t,a_t)}(z) \end{aligned} \tag{2.13}$$

where we have repeatedly used Property 2 of mixture distributions (Appendix A.2).

According to 2.13, $Z^\pi(s,a)$ can be interpreted as a convex combination of the sum of discounted reward distributions of all possible agent trajectories starting at the state-action (s,a) and following policy π from then on, each weight corresponding to the probability of that precise trajectory. It is important to note that Z^π encodes the intrinsic randomness of the agent's interactions with its environment; we should avoid considering it as a measure of uncertainty about the environment itself.

Moving to the *policy evaluation* setting, now we are interested in studying the behaviour of a distributional version of the policy evaluation operator defined in 2.4. Establishing the transition operator $P^\pi : \mathcal{Z} \to \mathcal{Z}$, where

$$P^\pi Z(s,a) \overset{D}{:=} Z(S',A'),$$

the *distributional Bellman operator* $\mathcal{T}^\pi : \mathcal{Z} \to \mathcal{Z}$ is defined as[4]

$$\mathcal{T}^\pi Z(s,a) \overset{D}{:=} R(s,a) + \gamma P^\pi Z(s,a); \tag{2.14}$$

We emphasize that three sources of randomness are involved in the compound distribution $\mathcal{T}^\pi Z$, i.e.

1. Randomness in the reward R,
2. Randomness in the transition P^π, and
3. Randomness in the next state-value distribution $Z(S',A')$,

[4]Note the abuse of notation, as the Bellman operator of value-based RL was also denoted by $\mathcal{T}^\pi$.

which together make this distributional Bellman operator fundamentally different to the expected value-cased one (Equation 2.4).

Now that we work in the space of distributions $\mathscr{P}(\mathbb{R})$, we note that convergence of distributional operators can also be assessed in terms of the contraction mapping theory, although it involves the non-trivial search of a metric over distributions on which the operator is a contraction. In the case of the distributional Bellman operator, though, some research has been already done on this task:

- The Kullback-Leibler (KL) divergence, one of the most widely used measures over distributions, is not an option for $\mathcal{T}^\pi$ to be a contraction mapping[25].
- In [17], authors demonstrate that $\mathcal{T}^\pi$ is a γ-contraction in the supremum-Wasserstein metric $\bar{d}_p$.
- In [25], it is proven that $\mathcal{T}^\pi$ is also a contraction in the supremum-Cramér metric $\bar{\ell}^2$, as well as in the d_π-weighted Cramér $\ell^2_{d_\pi}$.

Remark. We refer to the Appendix A.3 so see the definition of the mentioned measures/metrics over distributions.

Hence, provided we use the metrics $\bar{d}_p$, $\bar{\ell}^2$ or $\ell^2_{d_\pi}$, it can still be shown by applying Banach's Fixed-point theorem that the repeated application of the distributional Bellman operator -i.e. the process $Z_t = \mathcal{T}^\pi Z_{t-1}$, for some initial distribution Z_0- converges to the value distribution Z^π[17, 25].

However, more difficulties arise in the *control setting* when dealing with this distributional perspective; as stated in [17], while every optimal policy attain the same value Q^* in the expected-valued case, there might be many optimal value distributions. Considering the set Π^* of optimal policies, an optimal value distribution is defined as the value distribution of an optimal policy. Hence, the set of optimal value distributions is $\mathcal{Z}^* := \{Z^{\pi^*} : \pi^* \in \Pi^*\}$.

Note that an optimal value distribution must match the full distribution of returns under some optimal policy, so that not all value distributions with expectation Q^* are optimal. Under these conditions, the *distributional Bellman optimality operator* $\mathcal{T}$ does not have the same strong convergence guarantees of the policy evaluation operator[17, 25]: $\mathcal{T}$ is not a contraction in any usual metric between distributions, and its convergence to the set of optimal value distributions is weak.

2.3 Approximation Framework in the Distributional Setting

We should we aware that the full computation of the distributional Bellman operators on return distribution functions is generally either impossible (as we typically do not have access to the MDP dynamics, but to merely sample transitions) or infeasible (since the value distribution cannot be stored exactly in the general case).

This lead us to the design of several key approximations which are required to implement practical and scalable distributional RL algorithms[25]:

Distribution Parametrisation

Due to the fact that the full space of probability distributions, $\mathscr{P}(\mathbb{R})$, cannot be algorithmically encoded with a finite number of parameters, we need to approximate the distribution throughout a parametric family $\mathcal{P} \in \mathscr{P}(\mathbb{R})$.

Projection of Bellman Target Distribution

Another problem usually arises after computing the operator $\mathcal{T}^\pi$ -or its stochastic version- over a value distribution $Z_k \in \mathcal{P}^{\mathcal{S}\times\mathcal{A}}$: the new distribution may no longer lie in the selected parametric family $\mathcal{P}$. If this is the case, we further need to apply a *projection operator* $\Pi_M : \mathscr{P}(\mathbb{R})^{\mathcal{S}\times\mathcal{A}} \to \mathcal{P}^{\mathcal{S}\times\mathcal{A}}$, for some metric over distributions M, so as to map $\mathcal{T}^\pi Z_k$ into the proper parametric family, thus obtaining $\Pi_M \mathcal{T}^\pi Z_k$.

Notice that this forces us to consider the combined operator $\Pi_M \mathcal{T}^\pi$ for evaluating the convergence of the algorithm, and not simply $\mathcal{T}^\pi$. In particular, for the contraction mapping theory to apply, it is required to deal with a metric over distributions on which both Π_M and $\mathcal{T}^\pi$ are contractions. Therefore, the geometrical properties of the considered projection operator Π_M plays an important role in the obtained convergence guarantees.

Function Approximation

Distributional-based RL algorithms face the same problems than classical expected-based ones, so in many practical situations the large size of the involved state spaces prevents the use of tabular representations. Instead, they are also implemented using function approximation in order to estimate the return distribution of each state-action pair.

Whenever function approximation is used, however, we have the problem of having a limited set of representable value distributions. Analogously to what is done in expected-based RL with linear function approximations (Section 2.1), this can be faced by adding an extra projection step that takes any non-representable parametrized return distribution to the representable set $\mathcal{W}_\mathcal{P} := \{\hat{Z} \in \mathcal{P} : \hat{Z} \text{ is representable}\}$.

In practice, this implies applying another projection operator $\Pi_{M'} : \mathcal{P} \to \mathcal{W}_\mathcal{P}$, for a certain metric M', to the result of the combined operator $\Pi_M \mathcal{T}^\pi$, hence obtaining $\Pi_{M'} \Pi_M \mathcal{T}^\pi Z_k$. As one can infer, this adds more complexity in assessing the convergence guarantees of the algorithm; now we have to take care of the geometrical properties of all three involved operators.

Stochastic Bellman Operators

In order to evaluate the distributional Bellman operator $\mathcal{T}^\pi$, all possible next state-action-reward combinations should be taken into account. As in the expected valued case, the usual way of overcoming this practical limitation is by learning through transition samples (s, a, r, s', a') of the MDP. Hence, we can define a *stochastic distributional Bellman operator* $\widehat{\mathcal{T}}^\pi$ adapted to the randomness of these transitions, which defines a random measure whose behaviour is equal in expectation to the true Bellman operator $\mathcal{T}^\pi$.

Along our theoretical development, for the sake of simplicity, we will consider the simpler setting where stochasticity is not needed. Our analysis and described algorithms, however, can be easily extended to take into account stochastic operators.

Gradient Updates

Finally, having computed the estimate $\Pi_M \mathcal{T}^\pi Z_k$ (or $\Pi'_M \Pi_M \mathcal{T}^\pi Z_k$ if some function approximation is used) of the full target distribution, we still have to define how to compute the next iterate Z_{k+1}. In the tabular case, where an approximate parametrization distribution can be stored for each state-action pair, it can be simply defined as $Z_{k+1} = \Pi_M \mathcal{T}^\pi Z_k$, but

for the more general case where we use some function approximation to estimate value distributions the process is not that easy. In particular, as in expected-based RL with function approximation, the use of gradient updates seems to be appropriate in these cases[4], as it helps dissipate some noise introduced in the target by the stochastic approximation.

A key aspect here, however, is the not-straightforward (although by now expected) correspondence between the considered loss -which relates with $\Pi_{M'}$- and the projection Π_M; convergence and good behaviour of the resulting algorithm (i.e. contractibility of $\Pi'_M \Pi_M \mathcal{T}^\pi$) have only be proven when both of them rely on the same norm-induced geometry[25, 4].

2.4 Categorical Distributional Reinforcement Learning

Although the distributional perspective is almost as old as Bellman's equations[32], it was not until the recent introduction of the Categorical Distributional Reinforcement Learning[17] (CDRL) that it has become a central role within reinforcement learning. Their algorithm, called C51, was able to obtain state-of-the-art results in the Arcade Learning Environment[33] (ALE), outperforming the top expected-valued solutions by then.

Even though C51 is a clear heuristic-based example of a CDRL algorithm that lacks from any theoretical convergence guarantee, it can still be be fully characterized by the distributional RL framework described in the previous subsection:

- The 'C' of C51 accounts for 'categorical', which in turn comes from the distribution parametrisation that is used: the parametric family of categorical distributions over some fixed set of equally-spaced supports $z_1 < \cdots < z_K$:

$$\mathcal{P} = \left\{ \sum_{i=1}^{K} p_i \delta_{z_i} \middle| p_1, \ldots, p_K \geq 0, \sum_{k=1}^{K} p_k = 1 \right\}. \tag{2.15}$$

- It uses a non-linear function approximation for estimating the value distributions Z at each state-action pair; in particular, a neural network is used for that purpose.

- C51 learns from sampling transitions, so it implements the *stochastic distributional Bellman operator* $\widehat{\mathcal{T}}^\pi$ and the *stochastic distributional Bellman optimality operator* $\widehat{\mathcal{T}}^*$. Given a sampled transition $(s_t, a_t, r, s_{t+1}, a_{t+1})$, these stochastic operators basically transform the supports of the distributions by an affine shift map $f_{r,\gamma} : \mathbb{R} \to \mathbb{R}$, defined by $f_{r,\gamma}(z) = r + \gamma z$; in our notation,

$$\widehat{\mathcal{T}} Z_t(s_t, a_t) = (f_{r,\gamma})_\# Z_t(s_{t+1}, a_{t+1})$$

  with a_{t+1} either being selected by sampling the policy $\pi(\cdot|s_{t+1})$ (categorical policy evaluation, i.e. $\widehat{\mathcal{T}} = \widehat{\mathcal{T}}^\pi$), or being the action with the highest estimated expected returns (categorical Q-learning, $\widehat{\mathcal{T}} = \widehat{\mathcal{T}}^*$).

- It applies the heuristic projection operator Π_C after computing the stochastic Bellman operators in order to recover a distribution within the selected parametric family P. This projection is defined for single Dirac measures as

$$\Pi_C(\delta_y) = \begin{cases} \delta_{z_1} & y \leq z_1 \\ \frac{z_{i+1}-y}{z_{i+1}-z_i}\delta_{z_i} + \frac{y-z_i}{z_{i+1}-z_i}\delta_{z_{i+1}} & z_i < y < z_{i+1} \\ \delta_{z_K} & y > z_K \end{cases} \tag{2.16}$$

and can be easily extended to finite mixtures of Dirac measures as follows:

$$\Pi_C \left(\sum_{i=1}^{N} p_i \delta_{y_i} \right) = \sum_{i=1}^{N} p_i \Pi_C(\delta_{y_i})$$

The result of the projection step provides us with the target $\widehat{Z}_t(s_t, a_t) = \Pi_C \widehat{\mathcal{T}} Z_t(s_t, a_t)$.

- Finally, the original C51 performs a single step of gradient descent on the Kullback-Leibler divergence (see Appendix A.3 for the definition of this measure) of the prediction $Z_t(s_t, a_t)$ from the target $\widehat{Z}_t(s_t, a_t)$:

$$\mathrm{KL}\left(\widehat{Z}_t(s_t, a_t) || Z_t(s_t, a_t)\right)$$

with respect to the parameters of $Z_t(s_t, a_t)$; this gradient is used to generate the new estimate $Z_{t+1}(s_t, a_t) = \sum_{k=1}^{K} p_{t+1,k}(s_t, a_t)\delta_{z_k}$.

Pseudo-Algorithm 1 (provided by [25]) synthesises the steps performed by C51.

Algorithm 1: Categorical Distributional Reinforcement Learning: C51

Require: $Z_t(s, a) = \sum_{k=1}^{K} p_{t,k}(s, a)\delta_{z_k}$ for each (s, a)
Input: A sample transition (s_t, a_t, r_t, s_{t+1})
`#Compute distributional Bellman target`
if Categorical Policy Evaluation **then**
$a^* \sim \pi(\cdot|s_{t+1})$
else if Categorical Q-learning **then**
$a^* \leftarrow \arg\max_a Q(s_{t+1}, a)$
end if
$\widehat{Z}_*(s_t, a_t) \leftarrow (f_{r_t,\gamma})_{\#} Z_t(s_{t+1}, a^*)$
`#Project target onto support`
$\widehat{Z}_t(s_t, a_t) \leftarrow \Pi_C \widehat{Z}_*(s_t, a_t)$
`#Compute KL loss`
Find gradient $\mathrm{KL}\left(\widehat{Z}_t(s_t, a_t) || Z_t(s_t, a_t)\right)$
Generate new estimate $Z_{t+1}(s_t, a_t) = \sum_{k=1}^{K} p_{t+1,k}(s_t, a_t)\delta_{z_k}$
Output: Estimate $Z_{t+1}(s, a)$ for each (s, a)

Despite the good experimental results obtained by C51, all attempts to prove its convergence have failed so far. One of its most controversial steps, at least from a theoretical point of view, is the use of the KL divergence while applying the heuristic projection Π_C, as they clearly do not rely on the same norm-induced geometry[25].

We also recall that authors in [17] demonstrates, in the context of policy evaluation setting, that applying the Bellman operator T^π repeatedly to an initial return distribution function Z_0 guarantees convergence to the true set of return distributions Z^π in the supremum-Wasserstein metric.

However, after the introduction of the parametrization $\mathcal{P}$ and the projection operator Π_C, we have already noted that the operator to be analyzed is not the Bellman operator $\mathcal{T}^\pi$ itself anymore, but its composition with the projection operator, i.e. $\Pi_C \mathcal{T}^\pi$. As stated in [25], this can make contractivity break under all Wasserstein distances but $\bar{d}_1$:

Lemma 2.1. *$\Pi_C \mathcal{T}^\pi : \mathcal{Z} \to \mathcal{Z}$ is in general not a contraction in $\bar{d}_p$, for $p > 1$*

In contrast to that, Cramér distance seems to induce a useful geometric structure on the space of probability measures, allowing us to establish contractivity of the combined operator $\Pi_C \mathcal{T}^\pi$ in a much more natural way. We dedicate the following two sections to analyze CDRL algorithms together with Cramér-based metrics in the case of both Tabular Representations and Linear Function Approximations, respectively, which are the theoretical frameworks where stable behaviour and convergence properties are proven so far.

2.4.1 Tabular Representation

Our first approach to study Cramér-based CDRL algorithms will be to consider the simplest reinforcement learning framework, where we are able to store an approximate parametrization distribution for each state-action pair: the so-called tabular case.

Again, we will consider the parametric family $\mathcal{P}$ of categorical distributions over some fixed support $\{z_1, \dots, z_K\}$, as well as the heuristic projection operator Π_C defined in Equation 2.16 for mapping the backup distribution function $\mathcal{T}^\pi Z$ into $\mathcal{P}$. However, for the sake of simplicity and understandability, we will assume that no stochastic approximation is required.

First of all, we present the result of [25] that motivates the use of Cramér distance ℓ^2, which shows an interesting connection between this distance and the projection operator Π_C:

Proposition 2.2. *The Cramér metric ℓ^2 endows a particular subset of $\mathscr{P}(\mathbb{R})$ with a notion of orthogonal projection, and the orthogonal projection onto the subset $\mathcal{P}$ is exactly the heuristic projection Π_C. Consequently, Π_C is a non-expansion with respect to ℓ^2.*

The previous proposition directly leads to the contractivity of the operator $\Pi_C \mathcal{T}^\pi$ in the supremum-Cramér metric $\bar{l}_2$, which in turn ensures its convergence in the absence of stochastic approximation:

Proposition 2.3. *The operator $\Pi_C \mathcal{T}^\pi$ is a $\sqrt{\gamma}$-contraction in $\bar{\ell}^2$. Further, there is a unique distribution function $Z_C \in \mathcal{P}^{\mathcal{S}\times\mathcal{A}}$ to which the process $Z_{k+1} := \Pi_c \mathcal{T}^\pi Z_k$, given any initial distribution function $Z_0 \in \mathscr{P}(\mathbb{R})^{\mathcal{S}\times\mathcal{A}}$, converges in $\bar{\ell}^2$ as $k \to \infty$.*

Authors in [25] also address the natural question of how the limiting distribution function Z_C can differ from the true return distribution Z^π:

Proposition 2.4. *Let Z_C be the fixed point distribution of Proposition 2.3. If for all state-action pairs their corresponding true return distribution Z^π is supported on $[z_1, z_K]$, then*

$$\bar{\ell}^2(Z_C, Z^\pi) \leq \frac{1}{1-\gamma} \max_{1\leq i\leq K} (z_{i+1} - z_i).$$

Note that the true return distribution is gradually recovered as the fineness of the support increases. Hence, this result can be viewed as a way of quantifying the cost of using the parametrization $\mathcal{P}$ instead of fully non-parametric probability distributions. Finally, we also highlight the assumption that the support of the true return distributions lie on the selected support because we do not always have access to the scale of rewards in every RL problem; for these circumstances, an analogous result can be found in Proposition 4 of [25].

2.4.2 Linear Function Approximation

In this second and last theoretical approach, we go a step beyond the tabular case by considering a framework that makes use of function approximators to compute the parametrized

return distributions of each state-action pair, assuming a more realistic situation in which the state space is too large to store individual distributions.

We will restrict our analysis to the context of linear function approximation, whose convergence is already proven[26]. To the best of our knowledge, no Distributional RL algorithm with a non-linear function approximator has been proven to converge so far, despite the many efforts in studying C51.

Regarding the rest of approximations and assumptions of the CDRL algorithms considered in this study, they are exactly the same as in the previous Section 2.4.1: we contemplate the categorical parametric family $\mathcal{P}$ with fixed support $\{z_1, \dots, z_K\}$, the Cramér-based projection Π_C, and reject any stochastic approximation.

Under this linear regime, for the sake of simplicity, we redefine the approximated return distributions Z as mappings from states $s \in \mathcal{S}$ to vectors defined by a linear combination of features:

$$Z_\Theta(s) := \Theta^\top \phi(s) \tag{2.17}$$

where $\phi(s) \in \mathbb{R}^m$ is the feature vector at state s and $\Theta \in \mathbb{R}^{n \times K}$ represents the weight matrix. Therefore, we are assuming that the vector $Z_\Theta(s) \in \mathbb{R}^K$ is an estimation of a distribution over the support $\{z_1, \dots, z_K\}$, although it might have negative components and it is not necessarily normalized.

Notation. In vector notation, we will write $Z_\Theta = \Phi\Theta \in \mathbb{R}^{n \times K}$, where $\Phi \in \mathbb{R}^{n \times m}$ is the feature matrix, simply formed by all feature vectors.

Before continuing, let us carefully review the three different spaces of distributions-like objects that appear in CDRL with linear function approximation:

- On the one hand, there is the probability space where true return distributions lie; without any prior knowledge, one should simply consider the full space of distributions $Z^\pi \in \mathscr{P}(\mathbb{R})$. Assuming that it is known *a priori* the scale of the rewards, it can be reduced to the space of distributions with support in a certain interval; in our case, it could be $Z^\pi \in \mathscr{P}([z_1, z_K])$.
- On the other hand, the selected parametric family $\mathcal{P}$ of categorical distributions over the support $\{z_1, \dots, z_K\}$; so far, our estimated return distributions Z is expected to belong to this space, $Z \in \mathcal{P}$.
- Finally, and in addition to the previous two, there is now a vector space spanned by the features $\Phi \in \mathbb{R}^{n \times m}$, i.e. the set of representable return distributions $\mathcal{L} := \{\Phi\Theta : \Theta \in \mathbb{R}^{m \times K}\}$. This is, in fact, the crucial space in our development, as every approximated return distribution $\mathcal{Z}_\Theta$ lies in it. Ideally, if all Z_Θ were proper probability distributions over $\{z_1, \dots, z_K\}$, we would have $\mathcal{L} \subset \mathcal{P}$ (assuming some loss of expressiveness by our linear model).

Let us review the process. First, our linear function model provides us with return distribution estimates

$$Z_\Theta \in \mathcal{L}$$

for each state $s \in \mathcal{S}$. In order to get better approximations, CDRL algorithm applies the distributional Bellman operator, so we get

$$\mathcal{T}^\pi Z_\Theta \in \mathscr{P}([z_1, z_K]).$$

Then we would need to apply a first projection operator $\Pi_1 : \mathscr{P}([z_1, z_K]) \to \mathcal{P}$, for some metric M_1, so as to recover a distribution within our selected parametric family (like the Cramér projection Π_C), so

$$\Pi_1 \mathcal{T}^\pi Z_\Theta \in \mathcal{P}.$$

However, for the algorithm to compare distributions and improve its linear representations, now we need a further projection onto the space $\mathcal{L}$, which defines all the probability distributions that our linear model can actually represent. Thus, the idea is that a second projection operator $\Pi_2 : \mathcal{P} \to \mathcal{L}$, based on some metric M_2, is required so that

$$\Pi_2 \Pi_1 \mathcal{T}^\pi Z_\Theta \in \mathcal{L}.$$

As shown in [26], the (re)definition of both projections plays again a very important role for proving the convergence of the process $Z_{\Theta_{k+1}} = \Pi_2 \Pi_1 \mathcal{T}^\pi Z_{\Theta_k}$.

So far, it was demonstrated in the tabular case that the combined operator $\Pi_C \mathcal{T}^\pi$ is a contraction mapping in the supremum-Cramér metric $\bar{\ell}^2$; now the task is to achieve a similar result for the combination $\Pi_2 \Pi_1 \mathcal{T}^\pi$. For doing so, authors in [26] introduce a *generalized Cramér distance* which is able to deal with our return distribution estimates (we recall they are not necessarily probability distributions).

Let $C \in \mathbb{R}^{K \times K}$ denote the lower-triangular matrix of ones, and $e \in \mathbb{R}^K$ the vector of ones. Let $e_K = (1/\sqrt{K})e^\top$ and $\Pi_{e_K^\top} = I_K - e_K e_K^\top$. For any two discrete value distributions Z_1, Z_2 over the fixed support $\{z_1, \dots, z_K\}$, the generalized Cramér distance is defined as

$$\ell_\lambda^2(Z_1, Z_2) = (Z_1 - Z_2)^\top \Pi_{e_K^\top} C C^\top \Pi_{e_K^\top} (Z_1 - Z_2) + \lambda((Z_1 - Z_2)^\top e_K)^2 \tag{2.18}$$

The notation can be simplified by denoting $C_\lambda = \Pi_{e_K^\top} C C^\top \Pi_{e_K^\top} + \lambda e_K e_K^\top$, so the distance definition can be rewritten as

$$\ell_\lambda^2(Z_1, Z_2) = (Z_1 - Z_2)^\top C_\lambda (Z_1 - Z_2) = \|Z_1 - Z_2\|_{C_\lambda}$$

Looking at Equation 2.18, the idea is that the first term on the right-hand side penalizes the difference in cumulative probabilities of both distributions, whereas the second term accounts for the difference in mass; we refer to [26] for further details.

In addition to the Cramér distance generalization, authors in [26] also take into account the distribution ξ according to which the states to be updated are sampled; they use it to define a *ξ-weighted Cramér distance* over value distributions as

$$\ell_{\xi,\lambda}^2(Z_1, Z_2) := \sum_{s \in \mathcal{S}} \xi(s) \ell_\lambda^2(Z_1(s), Z_2(s)). \tag{2.19}$$

The two presented distances are precisely the key to all further results shown in [26], and therefore to prove the desired convergence of CDRL algorithm with linear function approximation.

In particular, the previously commented projections of value distributions Π_1 and Π_2 are defined through ℓ_λ^2 and $\ell_{\xi,\lambda}^2$, respectively. The former, which we will denote by $\Pi_{\lambda,\mathcal{P}}$, projects any value distribution Z onto the subspace $\mathcal{P}$ according to

$$\Pi_{\lambda,\mathcal{P}} Z := \arg\min_{Z' \in \mathcal{P}} \ell_\lambda^2(Z, Z'); \tag{2.20}$$

in fact, it can be seen that the previously considered Cramér projection Π_C satisfies this minimization expression[26]. The latter, which in our notation is expressed as $\Pi_{\xi,\lambda,\Phi}$, performs a ξ-weighted projection of any value distribution Z onto the set of value distributions Φ by:

$$\Pi_{\xi,\lambda,\Phi} Z := \arg\min_{\Phi\Theta, \Theta \in \mathbb{R}^{m \times K}} \ell_{\xi,\lambda}^2(Z, \Phi\Theta) \tag{2.21}$$

The following Lemma 2.5 shows an interesting result regarding the good behaviour of this projection $\Pi_{\xi,\lambda,\Phi}$

Lemma 2.5. *$\Pi_{\xi,\lambda,\Phi}$ is a non-expansion in $\ell^2_{\xi,\lambda}$; for every pair of return distributions Z, Z', we have*

$$\ell^2_{\xi,\lambda}(\Pi_{\xi,\lambda,\Phi}Z, \Pi_{\xi,\lambda,\Phi}Z') \leq \ell^2_{\xi,\lambda}(Z, Z')$$

At this point, before facing the proof of convergence of the whole CDRL algorithm, it is appropriate to rewrite distances ℓ^2_λ and $\ell^2_{\xi,\lambda}$ into two separate components: along dimension e_K and along the subspace A orthogonal to e_K, $A := \Pi_{e_K^\top} C$:

$$\begin{cases} \ell^2_\lambda(Z_1, Z_2) & = \|Z_1 - Z_2\|^2_{AA^\top} + \lambda\|Z_1 - Z_2\|^2_{e_K e_K^\top} \\ \ell^2_{\xi,\lambda}(Z_1, Z_2) & = \|Z_1 - Z_2\|^2_{\xi,AA^\top} + \lambda\|Z_1 - Z_2\|^2_{\xi,e_K e_K^\top} \end{cases} \tag{2.22}$$

The reason relies in Lemma 2.6, which states that the combined operator $\Pi_{\lambda,\mathcal{P}}\mathcal{T}^\pi$ behaves differently in each of these dimensions: while contracting all dimensions orthogonal to e_K -i.e. the subspace A- by a factor $\gamma^{1/2}$, it is only a non-expansion along e_K.

Lemma 2.6. *Let d_π the stationary distribution induced by policy π. For any two return distributions Z_1, Z_2, we have*

$$\begin{array}{rcl} \|\Pi_{\lambda,\mathcal{P}}\mathcal{T}^\pi Z_1 - \Pi_{\lambda,\mathcal{P}}\mathcal{T}^\pi Z_2\|^2_{d_\pi,AA^\top} & \leq & \gamma\|Z_1 - Z_2\|^2_{d_\pi,AA^\top} \\ \|\Pi_{\lambda,\mathcal{P}}\mathcal{T}^\pi Z_1 - \Pi_{\lambda,\mathcal{P}}\mathcal{T}^\pi Z_2\|^2_{d_\pi,e_K E_K^\top} & \leq & \|Z_1 - Z_2\|^2_{d_\pi,e_K E_K^\top} \end{array}$$

Note that, instead of the more general notation ξ, we are now specifying the stationary distribution d_π. This is simply due to the fact that in our CDRL algorithm we are actually following policy π to describe the agent trajectories we are learning from.

Finally, we present the theorem of [26] that provides us with the desired convergence guarantee for CDRL with Linear Function Approximation:

Theorem 2.7. *Let d_π the stationary distribution induced by policy π. The process*

$$Z_0 := \Phi\Theta_0, \quad Z_{k+1} := \hat{\Pi}_{d_\pi,\lambda,\Phi}T^\pi Z_k$$

converges to a set S such that, for any two $Z, Z' \in S$, there is a $\mathcal{S}$-indexed vector of constants α such that

$$Z(s) = Z'(s) + \alpha(s)e_K.$$

If $\lambda > 0$, S consists of a single point $\hat{Z}$ which is the fixed point of the process. Moreover, we can bound the error of this fixed point with respect to the true return distribution Z^π by

$$\ell^2_{d_\pi,\lambda}(\hat{Z}, Z^\pi) \leq \frac{1}{1-\gamma}\ell^2_{d_\pi,\lambda}(\Pi_{d_\pi,\lambda,\Phi}Z^\pi, Z^\pi) - \frac{\gamma\lambda}{1-\gamma}\|\hat{Z} - Z^\pi\|_{d_\pi,e_K e_K^\top},$$

where the second terms measures the difference in mass between $\hat{Z}$ and Z^π.

Note, however, that in order to prove convergence Theorem 2.7 do not make use of the projection $\Pi_{d_\pi,\lambda,\Phi}$, but the operator $\hat{\Pi}_{d_\pi,\lambda,\Phi}$, which is based on the loss

$$\hat{\ell}^2_\lambda(Z_1, Z_2) = (Z_1 - Z_2)^\top \Pi_{e_K^\top} CC^\top \Pi_{e_K^\top}(Z_1 - Z_2) + \lambda(Z_2 e - 1)^2 \tag{2.23}$$

instead of on the ℓ^2_λ distance. In fact, we refer to $\hat{\ell}^2_\lambda$ as a loss because it is not a distance, as it contains the explicit normalization penalty $(Z_2 e - 1)^2$ that encourages return distributions

to have unit mass. Given expression 2.23, the corresponding ξ-weighted Cramér loss $\hat{\ell}^2_{\xi,\lambda}$ is accordingly derived, and operator $\hat{\Pi}_{d_\pi,\lambda,\Phi}$ is defined so that, for a return distribution Z, it finds the value distribution in the span of Φ which minimizes $\hat{\ell}^2_{\xi,\lambda}(Z,\cdot)$:

$$\hat{\Pi}_{d_\pi,\lambda,\Phi} Z = \Phi\Theta^* \quad \text{where} \quad \Theta^* = \arg\min_\Theta \hat{\ell}^2_{\xi,\lambda}(Z, \Phi\Theta) \tag{2.24}$$

For bounding the approximation error we recover the well-defined distance $\ell^2_{d_\pi,\lambda}$, though.

As pointed out in [26], the parameter λ plays an important role in the theorem, both to guarantee convergence and to bound the approximation error. At a high level, this makes sense: a high value of λ forces the algorithm to output something close to a distribution, at the expense of actual predictions. On the other hand, taking $\lambda = 0$ yields a process which may not converge to a single point.

3 Covariate Shift Ratio

Before presenting our original work, we also need to introduce the Covariate Shift approach within Off-Policy Learning, which basically defines our starting point. In particular, this section is devoted to thoroughly review the Consistent Off-Policy Temporal Difference (COP-TD) solution presented in [27], as well as its discounted -and improved- version (Discounted COP-TD, or DCOP-TD), developed in [29].

To begin with, we describe the general Off-Policy Learning setting that will be used from this point on. Then, we assess the main theoretical concepts of COP-TD, paying special attention to its convergence guarantees together with linear function approximation. To conclude the section, we present the DCOP-TD approach and show how it enhances the aforementioned convergence guarantees of the undiscounted counterpart in practical applications.

3.1 Off-Policy Learning Setting

As stated in [29], now we move to the *policy evaluation* problem within *off-policy learning*, where we want to learn the value function V^π of a *target policy* π from samples drawn from P and a *behaviour policy* μ.

Remark. Up to this point, we only considered the *off-policy learning* setting, when π is both the behaviour and the target policy.

We first recall some notation:

- The Bellman equation for the state value function can be expressed in vector notation as $V^\pi = r_\pi + \gamma P_\pi V^\pi$, where $V^\pi \in \mathbb{R}^n$, $r_\pi \in \mathbb{R}^n$ and $P_\pi \in \mathbb{R}^{n\times n}$. The value function is in fact the fixed point of the *Bellman operator* $\mathcal{T}_\pi : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$, defined as $\mathcal{T}_\pi V := r_\pi + \gamma P_\pi V$. It defines a single step of *bootstrapping*: the process $V^{k+1} := \mathcal{T}_\pi V^k$ converges to V^π.
- Let $d \in \mathbb{R}^n$; we write $D_d \in \mathbb{R}^{n\times n}$ for the corresponding diagonal matrix, and consider the weighted squared seminorm notation of vectors $x \in \mathbb{R}^n$ $||x||_A^2 := ||Ax||^2 = x^T A^T A x$, $||x||_d^2 := ||x||_{D_d}^2 = \sum_{i=1}^n d(i)^2 x(i)^2$.
- $e \in \mathbb{R}^n$ accounts for the vector of all ones, and $\Delta(\mathcal{S})$ for the simplex over states: $d \in \Delta(\mathcal{S}) \implies d^T e = 1, d \geq 0$.
- We recall that $d \in \Delta(\mathcal{S})$ is the stationary distribution of a Markov Chain with transition function P if and only if $d = d \cdot P$. This distribution is unique when P defines a Markov chain with a single recurrent class[30].

In this particular setting we distinguish between two different state-to-state transition functions, P_π and P_μ, one for each policy; their respective stationary distributions will be represented by d_π and d_μ.

Linear Function Approximation

We recall from the linear RL setting described in Section 2.1 that the expected behaviour of the update rule (expression 2.7) is described by the projected Bellman operator $\Pi_d \mathcal{T}_\pi$, where Π_d represents the projection in the d-weighted L_2 norm ($d \in \Delta(\mathcal{S})$) onto the set of representable value functions by the considered linear model.

Moreover, we observed that the projection weights d must coincide with those of the stationary distribution of the target policy π for the combined operator to be a contraction (in particular, a contraction in the d_π-weighted L_2 norm).

However, in off-policy learning our available data is drawn from the behaviour policy μ, so states are considered to be updated according to d_μ instead of d_π. In particular, if nothing changes with respect to the on-policy setting, this implies that $\Pi_{d_\mu} \mathcal{T}^\pi$ is the actual expected behaviour of the implemented update rule 2.7.

One way of solving this consists in modifying the learning rule so that it can be considered under the sampling distribution d_π. An example of such a solution is the Covariate Shift Approach, explained in the following section.

3.2 Covariate Shift Approach

Supposing that stationary distributions d_π and d_μ are known, and that states are updated according to $s \sim d_\mu$, the covariate shift approach presented in [27] uses importance sampling to redefine 2.7 so that the semi-gradient update rule can be considered *under the sampling distribution* d_π:

$$\theta \leftarrow \theta + \alpha \frac{d_\pi(s)}{d_\mu(s)} \left[r + \gamma \phi(s')^T \theta - \phi(s)^T \theta \right] \phi(s) \tag{3.1}$$

with $a \sim \mu(\cdot|s)$, $r \sim R(s,a)$ and $s' \sim P(\cdot|s,a)$.

Hence, the Consistent Off-Policy Temporal Difference[27] (COP-TD) algorithm seek to learn that covariate shift ratio d_π/d_μ from samples by bootstrapping from a previous prediction, similar to temporal difference learning. Given a step size $\alpha > 0$, a ratio vector $c \in \mathbb{R}^n$ and a sample transition $(s_t, a_t, s_{t+1}) = (s, a, s')$ drawn from d_μ, $\mu(\cdot|s)$ and $P(\cdot|s,a)$, respectively, the COP-TD update is

$$c(s') \leftarrow c(s') + \alpha \left[\frac{\pi(a|s)}{\mu(a|s)} c(s) + c(s') \right] \tag{3.2}$$

The expected behaviour of this learning rule, which learns "in reverse" compared to TD learning, is captured by the *COP operator* Y:

$$(Yc)(s') := \mathop{\mathbb{E}}_{s \sim d_\mu, a \sim \mu(\cdot|s)} \left[\frac{\pi(a|s)}{\mu(a|s)} c(s) \middle| s' \right] \tag{3.3}$$

Note that the condition $s_{t+1} = s'$ in the expectation of 3.3 forces to take into account the distribution of previous state-action pairs (s,a) according to policy μ. The distribution of the possible previous states s is given by the time-reversal transition function $\bar{P}_\mu$, whose entries are:

$$\begin{aligned} \bar{P}_\mu(s|s') &:= Prob_\mu(s_t = s | s_{t+1} = s') \\ &= \frac{Prob_\mu(s_{t+1} = s' | s_t = s) Prob_\mu(s_t = s)}{Prob_\mu(s_{t+1} = s')} \\ &= \frac{P_\mu(s'|s) d_\mu(s)}{d_\mu(s')} \end{aligned} \tag{3.4}$$

Or, equivalently, $\bar{P}_\mu = D_{d_\mu}^{-1} P_\mu^T D_{d_\mu}$ in vector notation. Regarding the distribution of the possible actions that lead to s' from a certain state s by following policy μ, it will be represented

by function $\bar{\mu}$:

$$\begin{aligned}\bar{\mu}(a|s,s') &:= Prob_\mu(a_t = a|s_t = s, s_{t+1} = s') \\ &= \frac{Prob_\mu(a_t = a, s_t = s, s_{t+1} = s')}{Prob_\mu(s_t = s, s_{t+1} = s')} \\ &= \frac{Prob_\mu(s_{t+1} = s'|a_t = a, s_t = s)Prob_\mu(a_t = a|s_t = s)Prob_\mu(s_t = s)}{Prob_\mu(s_{t+1} = s'|s_t = s)Prob_\mu(s_t = s)} \\ &= \frac{P(s'|s,a)\mu(s|a)}{P_\mu(s'|s)}\end{aligned} \tag{3.5}$$

Bearing in mind the introduced notation, the expectation in 3.3 can be rewritten and expanded:

$$\begin{aligned}\mathop{\mathbb{E}}_{s\sim d_\mu, a\sim\mu(\cdot|s)}\left[\frac{\pi(a|s)}{\mu(a|s)}c(s)\middle|s'\right] &= \mathop{\mathbb{E}}_{s\sim\bar{P}_\mu(\cdot|s'), a\sim\bar{\mu}(\cdot|s,s')}\left[\frac{\pi(a|s)}{\mu(a|s)}c(s)\right] \\ &= \sum_s \bar{P}_\mu(s|s')\sum_a \bar{\mu}(a|s,s')\frac{\pi(a|s)}{\mu(a|s)}c(s) \\ &= \sum_s \left(\frac{P_\mu(s'|s)d_\mu(s)}{d_\mu(s')}\right)\sum_a \left(\frac{P(s'|s,a)\mu(s|a)}{P_\mu(s'|s)}\right)\frac{\pi(a|s)}{\mu(a|s)}c(s) \\ &= \frac{1}{d_\mu(s')}\sum_s d_\mu(s)c(s)\sum_a \pi(a|s)P(s'|s,a) \\ &= \frac{1}{d_\mu(s')}\sum_s P_\pi(s'|s)d_\mu(s)c(s)\end{aligned} \tag{3.6}$$

Thus, according to the last term of 3.6, the COP operator Y can be expressed in vector notation as

$$Yc = D_{d_\mu}^{-1}P_\pi^T D_{d_\mu}c \tag{3.7}$$

Regarding the behaviour of the operator, the following result of [29] guarantees that the process $c^{k+1} := Yc^k$ converges, and that any multiple of d_π/d_μ is a fixed point:

Theorem 3.1. *Suppose that P_π defines an ergodic Markov chain on $\mathcal{S}$, and let $c^0 \in \Delta(\mathcal{S})$. Then the process $c^{k+1} := Yc^k$ converges to $C\frac{d_\pi}{d_\mu}$, $C \in \mathbb{R}^+$.*

Furthermore, we can reduce the set of fixed points to the covariate shift ratio exclusively by considering a normalized version of the COP operator:

Corollary 3.2. *Suppose conditions of Theorem 3.1 are satisfied. Let*

$$(\bar{Y}c)(s') := \frac{(Yc)(s')}{\sum_s (Yc)(s)}$$

be the normalized COP operator. Then the unique fixed point of $\bar{Y}$ is the ratio d_π/d_μ, to which $c^{k+1} = \bar{Y}c^k$ converges.

COP-TD with Linear Function Approximation

Whenever the value function is approximated, we should expect to learn a ratio $\hat{c}$ estimate as well. In our analysis, we consider a linear approximation of the form

$$\hat{c}(s) = \phi(s)^\top w$$

where again ϕ defines a map from states to features, $\phi : \mathcal{S} \to \mathbb{R}^k$, and now $w \in \mathbb{R}^k$ is the vector of weights we are interested in learning. Note that in such a model negative ratio values are allowed; this can be avoid in practice by clipping those values at zero.

The introduced linear model, given a sample transition (s, a, s') drawn from d_μ, $\mu(\cdot|s)$ and $P(\cdot|s, a)$, respectively, induces the following semi-gradient update

$$\tilde{w} \leftarrow w + \alpha \left[\frac{\pi(a|s)}{\mu(a|s)} \phi(s)^\top w - \phi(s')w \right] \phi(s'), \tag{3.8}$$

which can be thought as a d-weighted projection Π_d for some $d \in \Delta(\mathcal{S})$[29]. However, an additional step is required for granting that the resulting ratio estimate $\hat{c}$ corresponds to some proper distribution ratio d/d_μ for $d \in \Delta(\mathcal{S})$. This is solved in [27] by following the update rule by a projection onto the d_μ-weighted simplex Δ_{Φ,d_μ}

$$w \leftarrow \arg\min_{u \in \mathcal{W}_{\Phi,d_\mu}} \|u - \tilde{w}\| \tag{3.9}$$

where $\mathcal{W}_{\Phi,d_\mu} := \{u \in \mathbb{R}^k : \sum_{s \in \mathcal{S}} d_\mu(s)\phi(s)^\top u = 1, \phi(s)^\top u \geq 0\}$. Looking at the definition of $\mathcal{W}_{\Phi,d_\mu}$, we can identify this second projection $\Pi_{\Delta_{\Phi,d_\mu}}$ as a normalization step as well.

The following Lemma 3.3 of [29] shows that, in fact, the normalization component of operator $\Pi_{\Delta_{\Phi,d_\mu}}$ is not only a convenience but also a requisite to get a good convergence guarantee.

Lemma 3.3. *Let Y be a symmetric COP-TD operator and Π the projection onto Φ in L_2 norm. If d_π/d_μ is not in the span of Φ, then $c = 0$ is the only fixed point of the process $c^{k+1} = \Pi Y c^k$.*

Hence, in order to get a meaningful approximation ratio, we must consider the repeated application of the combined operator $\Pi_{\Delta_{\Phi,d_\mu}} \Pi_d Y$, so that iterating $\hat{c}^{k+1} := \Pi_{\Delta_{\Phi,d_\mu}} \Pi_d Y \hat{c}^k$ provides the estimate.

In practice, however, that combination of the COP operator with the projection onto the d_μ-weighted simplex can be really hard to implement; authors in [27] presented a method for approximating the projection step in an online, sample-based setting for linear function approximation, but to the best of our knowledge there are no analogous implementation designs for other kind of function approximations, like neural networks.

3.3 Discounted COP-TD

Despite the good properties shown of COP-TD, there are two practical limitations of the algorithm that specially motivated the authors in [29] to work in an improvement. On the one hand, we have the previously commented difficulties to practically implement the projection operator $\Pi_{\Delta_{\Phi,d_\mu}}$ with function approximations other than linear ones. On the other hand, we note that the COP operator Y lacks a result ensuring it is a contraction mapping, which can make the algorithm converge at a slow rate or with high variance, or even be unstable together with function approximation.

They address these two issues in [29] through the *$\hat{\gamma}$-discounted COP-TD* learning rule; given a step size $\alpha > 0$, a ratio vector $c \in \mathbb{R}^n$ and a sample transition $(s_t, a_t, s_{t+1}) = (s, a, s')$ drawn from d_μ, $\mu(\cdot|s)$ and $P(\cdot|s, a)$, respectively, it modifies the previous COP-TD update 3.2 by

$$c(s') \leftarrow c(s') + \alpha \left[\hat{\gamma} \frac{\pi(a|s)}{\mu(a|s)} c(s) + (1 - \hat{\gamma}) - c(s') \right] \tag{3.10}$$

Again, the expected behaviour of this rule is captured by the corresponding $\hat{\gamma}$-discounted COP operator $Y_{\hat{\gamma}}$

$$(Y_{\hat{\gamma}}c)(s') := \mathop{\mathbb{E}}_{s\sim d_\mu, a\sim\mu(\cdot|s)}\left[\hat{\gamma}\frac{\pi(a|s)}{\mu(a|s)}c(s) + (1-\hat{\gamma})\middle| s'\right] = \hat{\gamma}(Yc)(s') + (1-\hat{\gamma}), \tag{3.11}$$

where Y is the COP operator (3.3). In vector notation we have

$$Y_{\hat{\gamma}}c = \hat{\gamma}D_{d_\mu}^{-1}P_\pi^T D_{d_\mu}c + (1-\hat{\gamma})e \tag{3.12}$$

From the above expressions, it is straightforward to notice that we recover COP-TD for $\hat{\gamma} = 1$, i.e. $Y_1 = Y$.

So as to characterize the discounted COP operator authors in [29] introduce the discounted reset transition function $\hat{P}_\pi$, which for a given $\hat{\gamma} \in [0,1]$ is defined as

$$\hat{P}_\pi := \hat{\gamma}P_\pi + (1-\hat{\gamma})ed_\mu^\top,$$

denoting by $\hat{d}_\pi$ its stationary distribution (i.e. $\hat{d}_\pi = \hat{d}_\pi\hat{P}_\pi$). We can interpret that $\hat{P}_\pi$ encodes a stochastic process in which either transitions occur as usual with probability $\hat{\gamma}$, or resets to the stationary distribution d_μ with the remainder probability. Lemma 3.4 gives us more insights about how operator $Y_{\hat{\gamma}}$ is intimately connected to this process:

Lemma 3.4. *For $\hat{\gamma} < 1$, the ratio $\hat{d}_\pi/d_\mu$ is the unique fixed point of the operator $Y_{\hat{\gamma}}$, where $\hat{d}_\pi$ is the stationary distribution of the transition function $\hat{P}_\pi$ corresponding to the given $\hat{\gamma}$*

So, basically, the discounted reset transition function encodes the transition dynamics associated to the use of the DCOP operator. Furthermore, as shown in the following result of [29], we can also get convergence guarantees for the repeated application of $Y_{\hat{\gamma}}$ for $\hat{\gamma} < 1$ without requiring positive initial values or any normalization.

Theorem 3.5. *Given $\hat{\gamma} < 1$, the process $c^{k+1} := Y_{\hat{\gamma}}c^k$ converges to $\hat{d}_\pi/d_\mu$ for any $c^0 \in \mathbb{R}^n$.*

DCOP-TD with Linear Function Approximation

Now it is time to analyze how DCOP-TD algorithm behaves when it is combined with (linear) function approximation, and in particular whether it provides us with good convergence guarantees in a practical online application.

As in COP-TD, a linear ratio estimate $\hat{c}(s) = \phi(s)^\top w$ is considered, and sample transitions (s,a,s') are assumed to be drawn from d_μ, $\mu(\cdot|s)$ and $P(\cdot|s,a)$, respectively. This induces the semi-gradient update

$$\tilde{w} \leftarrow w + \alpha\left[\left(\hat{\gamma}\frac{\pi(a|s)}{\mu(a|s)}\phi(s)^\top w + (1-\hat{\gamma})e\right) - \phi(s')w\right]\phi(s'), \tag{3.13}$$

which again can be interpreted as a d-weighted projection for some $d \in \Delta(\mathcal{S})$. Nevertheless, now the additional, hard-to-implement projection-normalization step is not required for the process to converge. In this context, authors in [29] argue that $s' \sim d_\mu$ -since d_μ is the stationary distribution that ens up ruling the sampled agent trajectories-, and therefore that the induced d-weighted projection should be in fact considering $d = d_\mu$. As a result, the process we are analyzing can be described by the projected DCOP operator

$$\Pi_{d_\mu}Y_{\hat{\gamma}}$$

Before presenting any convergence result, however, we need to introduce the concentration coefficient $K_{\pi,\mu,n}$ defined in Lemma 3.6; at a high level, it measures the discrepancy in stationary distributions -$d\pi$ and d_μ- between a pair of states that can be considered 'close' according to policy π, in the sense that one of these states is reachable from the other in n steps; it simplifies to 1 when $\pi = \mu$.

Lemma 3.6. *The induced operator norm of the COP operator Y^n is upper bounded by a constant $\sqrt{K_{\pi,\mu,n}}$ in the sense that*

$$\|Y^n\|_{d_\mu}^2 \leq K_{\pi,\mu,n} := \sup_{s' \in \mathcal{S}} \sum_{s \in \mathcal{S}} \frac{d_\mu(s)}{d_\mu(s')} P_\pi^n(s'|s).$$

Further, the series can be bounded by a constant,

$$K_{\pi,\mu,n} \leq K_{\pi,\mu} := \left\| \frac{d_\mu(s)}{d_\pi(s)} \right\|_\infty \left\| \frac{d_\pi(s)}{d_\mu(s)} \right\|_\infty$$

Theorem 2.7 provides us with the reason why this concentration coefficient is required in our study: authors in [29] make use of it to obtain a safe value of $\hat{\gamma}$ below which the DCOP operator is a contraction mapping, and from this they can prove convergence of the combined operator $\Pi_{d_\mu} Y_{\hat{\gamma}}$ in the linear function approximation case.

Theorem 3.7. *Consider the DCOP operator $\hat{Y}_{\hat{\gamma}}$. For any $c \in \mathbb{R}^n$,*

$$\left\| \hat{Y}_{\hat{\gamma}} c - \frac{\hat{d}_\pi}{d_\mu} \right\|_{d_\mu} \leq \hat{\gamma}^n \sqrt{K_{\pi,\mu,1}} \left\| c - \frac{\hat{d}_\pi}{d_\mu} \right\|_{d_\mu}$$

and in particular $\hat{Y}_{\hat{\gamma}}$ is a contraction mapping for $\hat{\gamma} < (K_{\pi,\mu,n})^{-\frac{1}{2}}$. Since $K_{\pi,\mu,1}$ is a bounded series, the exponential factor is guaranteed to dominate, so there exists a value of $\hat{\gamma} < 1$ for which the projected DCOP operator $\Pi_{d_\mu} \hat{Y}_{\hat{\gamma}}$ is a contraction mapping.

Remark. We note that the actual Theorem 4 of paper [29], from which previous Theorem 3.7 is extracted, provides a slightly more general result for n-step updates; we refer to [1] for further details about these methods. For simplicity, we have reduced it to the case of 1-step updates, as we have been considering in the rest of our analysis.

So, provided we take a sufficiently small $\hat{\gamma}$, Theorem 3.7 guarantees that DCOP-TD overcomes the divergence issues of COP-TD by granting that $\Pi_{d_\mu} \hat{Y}_{\hat{\gamma}}$ is a contraction mapping. In addition, the empirical evaluation performed on [29] of this discounted version suggests that it is unlikely to be in the worst-case scenario achieving $K_{\pi,\mu,1}$ strictly, as the algorithm avoid divergence even with large $\hat{\gamma}$ values.

4 Distributional Covariate Shift Approach

Now that we have introduced both Distributional RL and the Covariate Shift Approach for Off-policy Learning, we are ready to present our original work.

Similarly to what was done in [17] by going beyond the notion of value within the reinforcement learning setting, we first wanted to analyze a distributional perspective of the covariate shift approach. Hence, our starting point could be the following distributional equation

$$X(s') \overset{D}{=} \frac{\pi(A^{\mu}_{s,s'}|S^{\mu}_{s'})}{\mu(A^{\mu}_{s,s'}|S^{\mu}_{s'})} X(S^{\mu}_{s'}) \tag{4.1}$$

where

- X is the random ratio between distributions of a certain state.
- $(S^{\mu}_{s'}, A^{\mu}_{s,s'})$ is the previous state-action random variable:
  - The random variable $S^{\mu}_{s'}$ represents the states from which state s' is achievable by following policy μ; $S^{\mu}_{s'} = s$ with probability $\bar{P}_{\mu}(s|s')$
  - $A^{\mu}_{s,s'}$ encodes the random action that can be taken to get state s' from a state $s \sim S^{\mu}_{s'}$ according to policy μ, so $A^{\mu}_{s,s'} = a$ with probability $\bar{\mu}(a|S^{\mu}_{s'}, s')$

Thus, paying attention to equation 4.3, we note that it intrinsically expresses the random ratio of a state $X(s_{t+1})$ as a mixture distribution with the 'corrected' previous state random ratios $\frac{\pi(a_t|s_t)}{\mu(a_t|s_t)}X(s_t)$ as mixing components, and the previous state-action random variable $(S^{\mu}_{s_{t+1}}, A^{\mu}_{s_t,s_{t+1}})$ as the mixing distribution (see Appendix A.2 for more details about mixture distributions):

$$f_{X(s_{t+1})}(x) = \sum_{s_t} \bar{P}_{\mu}(s_t|s_{t+1}) \sum_{a_t} \bar{\mu}(a_t|s_t, s_{t+1}) f_{\frac{\pi(s_t,a_t)}{\mu(s_t,a_t)}X(s_t)}(x) \tag{4.2}$$

Notation. So as to reduce the complexity and increase the readability of the formulation, we introduce the following notation:

- Let define ρ the policy ratio π/μ:

$$\rho(a, s) := \frac{\pi(a|s)}{\mu(a|s)}$$

- Note in equation 4.2 that there are as many mixture components as state-action pairs (s_t, a_t); thus, we can iterate the summation over all these possible pairs and define each corresponding mixture weight α as

$$\alpha(s_t, a_t|s_{t+1}) = \bar{P}_{\mu}(s_t|s_{t+1})\bar{\mu}(a_t|s_t, s_{t+1})$$

4.1 Distributional DCOP-TD

We will directly develop the basis of a Distributional Discounted COP-TD, which has provided us with better convergence guarantees in the expected-valued case. Thus, the considered distributional equation becomes

$$X(s') \overset{D}{=} \hat{\gamma}\rho(S^{\mu}_{s'}, A^{\mu}_{s,s'})X(S^{\mu}_{s'}) + 1 - \hat{\gamma}, \tag{4.3}$$

where $\hat{\gamma} \in (0, 1)$ is the discount factor, and X represents now the random discounted ratio between distributions at a certain state.

Definition 1. We define the *distributional DCOP operator* $Y^D_{\hat{\gamma}} : \mathscr{P}(\mathbb{R}) \to \mathscr{P}(\mathbb{R})$ as

$$(Y^D_{\hat{\gamma}} X)(s_{t+1}) :\overset{D}{=} \hat{\gamma}\ \rho(S^{\mu}_{s_{t+1}}, A^{\mu}_{s_t,s_{t+1}}) X(S^{\mu}_{s_{t+1}}) + 1 - \hat{\gamma}$$

so that

$$f_{(Y^D X)(s_{t+1})}(x) = \sum_{(s_t,a_t)} \alpha(s_t, a_t|s_{t+1}) \cdot f_{\hat{\gamma}\rho(s_t,a_t)X(s_t)+1-\hat{\gamma}}(x)$$

In addition, we can recover the distributional undiscounted setting by simply considering $\hat{\gamma} = 1$, with the corresponding *distributional COP operator* $Y^D = Y^D_1$.

Lemma 4.1. *Let $(Y^D_{\hat{\gamma}} X)(s) \in \mathscr{P}(\mathbb{R})$ be the resulting distribution of applying the distributional DCOP operator over a random discounted ratio $X(s)$, for any state $s \in \mathcal{S}$. Then we have that*

$$\mathbb{E}[(Y^D_{\hat{\gamma}} X)(s)] = Y_{\hat{\gamma}}(\mathbb{E}[X(s)])$$

where $Y_{\hat{\gamma}}$ is the original value-based DCOP operator defined in Equation 3.11.

Proof. Let's consider $s_{t+1} \in \mathcal{S}$. The result follows from expanding the expectation of $(Y^D_{\hat{\gamma}} X)(s_{t+1})$ by using the properties of Mixture Distributions (Appendix A.2):

$$\begin{aligned}
\mathbb{E}[(Y^D_{\hat{\gamma}} X)(s_{t+1})] &= \int_{-\infty}^{\infty} x f_{(Y^D_{\hat{\gamma}} X)(s_{t+1})}(x) dx \\
&= \sum_{(s_t,a_t)} \alpha(s_t, a_t|s_{t+1}) \cdot \int_{-\infty}^{\infty} x f_{\hat{\gamma}\rho(s_t,a_t)X(s_t)+1-\hat{\gamma}}(x) dx \\
&= \sum_{(s_t,a_t)} \alpha(s_t, a_t|s_{t+1}) \cdot \mathbb{E}\left[\hat{\gamma}\rho(s_t, a_t)X(s_t) + 1 - \hat{\gamma}\right] \\
&= 1 - \hat{\gamma} + \hat{\gamma} \sum_{(s_t,a_t)} \alpha(s_t, a_t|s_{t+1}) \cdot \rho(s_t, a_t) \cdot \mathbb{E}\left[X(s_t)\right] \\
&= 1 - \hat{\gamma} + \hat{\gamma} \mathop{\mathbb{E}}_{s_t \sim \bar{P}_{\mu}(\cdot|s_{t+1}), a_t \sim \bar{\mu}(\cdot|s_t,s_{t+1})} [\rho(s_t, a_t)\, \mathbb{E}[X(s_t)]] \\
&= 1 - \hat{\gamma} + \hat{\gamma} Y(\mathbb{E}[X(s)]) = Y_{\hat{\gamma}}(\mathbb{E}[X(s)])
\end{aligned} \tag{4.4}$$

where the last but one step holds taking into account the second term of the expansion 3.6 of the COP operator Y. □

In the previous Lemma 4.1 we show that the expectation of the learned ratio distributions coincide with the values we would learn by applying the usual DCOP operator to the expected values of the ratio distributions. This helps us prove the following Corollary:

Corollary 4.2. *Consider the process $X^{k+1} := Y^D_{\hat{\gamma}} X^k$ for any initial ratio distribution $X^0 \in \mathcal{P}(\mathbb{R})^n$. Then $\mathbb{E}[X^k]$ converges to some fixed point $c \in \mathbb{R}^n$ as $k \to \infty$.*

Proof. Lemma 4.1 tells us that:

$$\mathbb{E}[X^{k+1}] = \mathbb{E}[Y^D_{\hat{\gamma}} X^k] = Y_{\hat{\gamma}}(\mathbb{E}[X^k])$$

Defining $c^k := \mathbb{E}[X^k]$, we note that the above expression defines the process

$$c^{k+1} = Y_{\hat{\gamma}} c^k$$

Invoking the convergence result (Theorem 3.5) of the DCOP operator $Y_{\hat{\gamma}}$, we conclude that $c^k = \mathbb{E}[X^k]$ converges to some fixed point $c \in \mathbb{R}^n$. □

Note that Corollary 4.2 allows us to estimate the Covariate Shift Ratio values in this Distributional DCOP-TD setting as simply

$$\frac{d_\pi}{d_\mu}(s) = \mathbb{E}[X(s)]$$

However, so far we have not shown any convergence guarantee for the iterates $\{X_k\}$ to the true ratio distribution X. That is precisely what motivated us to develop the following logarithmic approach, which indirectly provides us with the desired distributional convergence of the $Y^D_{\hat{\gamma}}$ operator.

4.2 Log-Distributional Covariate Shift Approach

Note that the Distributional Covariate Shift Equation 4.3 is purely multiplicative, and so it is the associated update rule in the learning setting. This complicates the analysis, possibly being one of the reasons why we could not find any convergence guarantee for the distributional DCOP-TD algorithm.

There was, though, one way of transforming that multiplicative behaviour into a better-suited aditive one: working in the logarithmic space. To do so, we first define the corresponding log-ratio distributions:

Definition 2. Consider any state $s_{t+1} \in \mathcal{S}$, and the corresponding ratio distribution $X(s_{t+1})$. We define the log-ratio distribution $W(s_{t+1}) \in \mathscr{P}(\mathbb{R})$ as

$$W(s_{t+1}) := \log\left(X(s_{t+1})\right),$$

so that

$$\begin{aligned} f_{W(s_{t+1})}(x) &= \sum_{(s_t,a_t)} \alpha(s_t, a_t|s_{t+1}) \cdot f_{\log(\rho(s_t,a_t)X(s_t))}(x) \\ &= \sum_{(s_t,a_t)} \alpha(s_t, a_t|s_{t+1}) \cdot f_{\log(\rho(s_t,a_t))+W(s_t)}(x) \end{aligned}$$

Having these new distribution objects lying in the logarithmic space, our proposal is to learn an estimate of the true log-ratio distribution W^C in that space. Hence, in the undiscounted case, the corresponding learning rule would be captured by the following operator:

Definition 3. We define the Log-Distributional COP operator $G : \mathscr{P}(\mathbb{R}) \to \mathscr{P}(\mathbb{R})$ as

$$(GW)(s_{t+1}) := \log\left(\rho(S^\mu_{s_{t+1}}, A^\mu_{s_t,s_{t+1}})\right) + W(S^\mu_{s_{t+1}}) \tag{4.5}$$

so that

$$f_{(GW)(s_{t+1})}(x) = \sum_{(s_t,a_t)} \alpha(s_t, a_t|s_{t+1}) \cdot f_{\log(\rho(s_t,a_t))+W(s_t)}(x)$$

Remark. At this point, one might wonder why this logarithmic approach was not considered in the original expected-valued (D)COP-TD development; the reason is that, in general, Jensen's inequality prevents the correspondence between the multiplicative fixed point $c \in \mathbb{R}^n$ and the exponential of the log fixed point:

$$\mathbb{E}[c(s)] \neq \exp(\mathbb{E}[\log c(s)]) \quad \text{since} \quad \mathbb{E}[\log c(s)] \leq \log(\mathbb{E}[c(s)])$$

In contrast to that, in the distributional setting the distribution mixture plays the role of the expectation in the valued case, and we do can interchange mixtures and functions thanks to their properties (see Appendix A.2):

$$X(s) = \exp(\log(X(s)) \tag{4.6}$$

Note that previous expression 4.6 ensures that learning the additive fixed distribution W^C in log space and exponentiating it would lead to the proper ratio distribution. However, we emphasize that, so far, we do not have any guarantee that these distributional fixed points exist, neither in log space nor in the original multiplicative one.

What we do know, though, is that the expectation of the ratio distributions that result from the repeated application of the distributional (D)COP operator converge (Corollary 4.2); the following Proposition 4.3 shows that this result can be extended to exponentiated log-ratio distributions, so that, again, we can estimate the Covariate Shift Ratio values in the Log-Distributional setting as simply

$$\frac{d_\pi}{d_\mu}(s) = \mathbb{E}[\exp(W(s))]$$

Proposition 4.3. *Consider the process $W^{k+1} := GW^k$ for any initial log-ratio distribution $W^0 \in \mathscr{P}(\mathbb{R})^n$. Then $\mathbb{E}[\exp(W^k)]$ converges to some fixed point $c \in \mathbb{R}^n$ as $k \to \infty$.*

Proof. In particular, by the definition of a log-ratio distribution, the sequence $\{W^k\}$ induces a sequence of ratio distributions $\{X^k\}$, $X^k \in \mathscr{P}(\mathbb{R})^n$, such that, for every state $s_{t+1} \in \mathcal{S}$,

$$\begin{aligned} X^{k+1}(s_{t+1}) &= \exp(W^{k+1}(s_{t+1})) = \exp(GW^k(s_{t+1})) = \exp(\log\left(\rho(S^\mu_{s_{t+1}}, A^\mu_{s_t,s_{t+1}})\right) + W(S^\mu_{s_{t+1}})) \\ &= \exp(\log\left(\rho(S^\mu_{s_{t+1}}, A^\mu_{s_t,s_{t+1}})X(S^\mu_{s_{t+1}})\right) = Y^D X^k(s) \end{aligned}$$

In compact notation,

$$X^{k+1} = \exp(W^{k+1}) = \exp(GW^k) = Y^D X^k$$

From that, the result follows by taking expectations and applying Corollary 4.2. □

Our goal, nonetheless, is to get a more robust convergence result beyond expectations: we would like to prove the existence -and uniqueness- of some fixed point W^C. In practice, as we have shown in previous Sections, the discounted version is potentially more suitable to attain that objective, but now that we work in a logarithmic space we face the problem of how exactly applying the $\hat{\gamma}$-discounting in order to get a practical and reasonable algorithm.

We finally implemented the discounting so that the resulting operator belongs to the class of Bellman operators already detailed and analyzed in the RL literature. As we will see, this will greatly simplify our study, and will provide us with strong convergence results.

Definition 4. We define the Log-Distributional DCOP operator $G_{\hat{\gamma}} : \mathscr{P}(\mathbb{R}) \to \mathscr{P}(\mathbb{R})$ as

$$(G_{\hat{\gamma}}W)(s_{t+1}) := \log\left(\rho(S^\mu_{s_{t+1}}, A^\mu_{s_t,s_{t+1}})\right) + \hat{\gamma}W(S^\mu_{s_{t+1}}) \tag{4.7}$$

so that

$$f_{(G_{\hat{\gamma}}W)(s_{t+1})}(x) = \sum_{(s_t,a_t)} \alpha(s_t, a_t|s_{t+1}) \cdot f_{\log(\rho(s_t,a_t))+\hat{\gamma}W(s_t)}(x)$$

The similarities of $G_{\hat{\gamma}}$ with the usual distributional Bellman operator (Expression 2.14) are quite evident; in fact, the Log-Distributional DCOP operator defines a distributional Bellman update where

- the policy that drives the agent trajectory is μ
- the state-to-state transition function is $\bar{P}_\mu$, the time reversal transition function defined in 3.4.
- the immediate reward is $\log\left(\rho(S^\mu_{s_{t+1}}, A^\mu_{s_t,s_{t+1}})\right)$
- the discount factor is $\hat{\gamma}$
- the value distribution W is expressed as a sum of $\hat{\gamma}$-discounted rewards

The benefits of identifying $G_{\hat{\gamma}}$ as a Bellman operator are hugely relevant, since all results previously presented of the Bellman operator $\mathcal{T}^\pi$ of Distributional Reinforcement Learning hold for our Log-Distributional DCOP operator.

Hence, we can at last prove the desired convergence of log-ratio distributions to some fixed point in the general distributional setting when using the supremum-Wasserstein metric:

Lemma 4.4. *$G_{\hat{\gamma}} : \mathcal{P}(\mathbb{R}) \to \mathcal{P}(\mathbb{R})$ is a $\hat{\gamma}$-contraction in $\bar{d}_p$.*

Proof. Proof of Lemma 3 of [17]. □

Corollary 4.5. *The process $W^{k+1} := G_{\hat{\gamma}} W^k$, for any initial log-ratio distribution W^0, converges to W^C in $\bar{d}_p$.*

Proof. Using Lemma 4.4, we conclude applying Banach's fixed point theorem that $G_{\hat{\gamma}}$ has a unique fixed point W^C. As we assume all moments are bounded, this is sufficient to conclude that the sequence $\{W^k\}$ converges to W^C in $\bar{d}_p$ for $1 < p < \infty$. □

Remark. Indirectly, convergence of ratio distributions has been also proven: any sequence of log-distributional ratios generates a sequence of distributional ratios simply by their definition, and the shown convergence of the former implies the convergence of the latter. However, as the discounting implementation of operator $G_{\hat{\gamma}}$ in the logarithmic setting does not exactly mathematically correspond to that of $Y^D_{\hat{\gamma}}$ in the multiplicative case, there is no an exact connection between these two operators, which means we do not have a clear convergence result to show.

4.3 Categorical Log-Distributional DCOP-TD

The idea is to design a Categorical framework for Log-Distributional DCOP-TD analogous to those of CDRL algorithms whose convergence has been proven. Hence, we will make use of the following approximations:

- We consider the parametric family of categorical distributions over some fixed set of equally-spaced supports $w_1 < \cdots < w_K$:

$$\mathcal{P} = \left\{ \sum_{i=1}^{K} p_i \delta_{w_i} \middle| p_1, \ldots, p_K \geq 0, \sum_{k=1}^{K} p_k = 1 \right\}$$

- For any state $s_{t+1} \in \mathcal{S}$, we assume $\rho(S^{\mu}_{s_{t+1}}, A^{\mu}_{s_t,s_{t+1}})$ can be computed, and we will denote this value as simply ρ.

- Given a state $s_{t+1} \in \mathcal{S}$ and its corresponding ρ, the Log-Distributional DCOP operator $G_{\hat{\gamma}}$ transforms the support $\{w_1, \dots, w_K\}$ of log-ratio distributions by an affine shift map $f_{\log(\rho),\hat{\gamma}} : \mathbb{R} \to \mathbb{R}$ defined by $f_{\log(\rho),\hat{\gamma}}(w_i) := \log(\rho) + \hat{\gamma} w_i$. Hence, we can write

$$G_{\hat{\gamma}} W(s) := (f_{\log(\rho),\hat{\gamma}})_{\#} W(s)$$

- We apply the heuristic projection operator Π_C defined in 2.16 so as to recover a ratio distribution within $\mathcal{P}$ after applying the Log-Distributional DCOP operator $G_{\hat{\gamma}}$.

- We denote by $W^{\mu} \in \mathscr{P}(\mathbb{R})$ the true log-ratio distribution which we aim to estimate.

In the following subsections we present the corresponding convergence results for Categorical Log-Distributional DCOP-TD algorithms when using Tabular Representation and Linear Function Approximation, respectively.

4.3.1 Tabular Representation

Let us first consider the simplest setting, where we assume we are able to store an approximate parametrized log-ratio distribution for each state, and no stochastic approximation is required during the learning process. Given the above categorical framework, the operator to be analyzed is now $\Pi_C G_{\hat{\gamma}}$.

However, due to the fact that $G_{\hat{\gamma}}$ is a Bellman operator, we can re-use the results of Section 2.4.1 for CDRL algorithms with Tabular Representation. This dramatically simplifies our study, as we can directly show a strong convergence result in the supremum-Cramér distance:

Proposition 4.6. *The operator $\Pi_C G_{\hat{\gamma}}$ is a $\sqrt{\hat{\gamma}}$-contraction in $\bar{\ell}^2$. Further, there is a unique distribution function $W^C \in \mathcal{P}^{\mathcal{S}}$ to which the process $W^{k+1} := \Pi_c G_{\hat{\gamma}} W^k$, given any initial log-ratio distribution function $W^0 \in \mathscr{P}(\mathbb{R})^{\mathcal{S}}$, converges in $\bar{\ell}^2$ as $k \to \infty$.*

Proof. Proof of Proposition 2.3. □

Moreover, when we know a priori that log-ratio distributions lie in a certain interval, we can easily bound the error of our estimate W^C with respect to the true log-ratio distribution W^{μ}:

Proposition 4.7. *Let W^C be the limiting return distribution of Proposition 4.6. If for all states their corresponding true log-ratio distribution W^{μ} is supported on $[w_1, w_K]$, then*

$$\bar{\ell}^2(W^C, W^{\mu}) \leq \frac{1}{1-\hat{\gamma}} \max_{1 \leq i \leq K} (w_{i+1} - w_i).$$

Proof. Proof of Proposition 2.4. □

This difference can be interpreted as the cost of using the parametrization $\mathcal{P}$; we observe how W^{μ} can be recovered by increasing the fineness of the support.

4.3.2 Linear Function Approximation

In our second and last framework, we consider the use of linear function approximation to estimate the parametrized log-ratio distribution of each state. We also make use of the Categorical setting described above, and avoid any stochasticity in the following analysis.

We will introduce the same notation that was presented in Section 2.4.2 for CDRL with Linear Function Approximation. Thus, the linear model redefines the approximated log-ratio distributions W as mappings from states $s \in \mathcal{S}$ to vectors defined by a linear combination of features:

$$W_\Theta(s) := \Theta^\top \phi(s) \tag{4.8}$$

where $\phi(s) \in \mathbb{R}^m$ is the feature vector at state s and $\Theta \in \mathbb{R}^{n\times K}$ represents the weight matrix. In vector notation, we have $W_\Theta = \Phi\Theta \in \mathbb{R}^{n\times K}$, where $\Phi \in \mathbb{R}^{n\times m}$ is the feature matrix.

Remark. We recall the assumption that vector $W_\Theta(s) \in \mathbb{R}^K$ is an estimation of a distribution over the support $\{w_1, \dots, w_K\}$, although it might have negative components and it is not necessarily normalized.

Again, all the theory developed in Section 2.4.2 with Linear Function Approximation is valid for our Categorical Log-Distributional DCOP-TD due to the class equivalence of operators $G_{\hat\gamma}$ and $\mathcal{T}^\pi$, and the equivalence of the Categorical framework. So, in order not to repeat step by step the study presented in that Section 2.4.2, we can directly show the final convergence result that applies in our case:

Theorem 4.8. *Let d_μ the stationary distribution induced by policy μ. The process*

$$W^0 := \Phi\Theta^0, \quad W^{k+1} := \hat\Pi_{d_\mu,\lambda,\Phi} G_{\hat\gamma} W^k$$

converges to a set S such that, for any two $W, W' \in S$, there is a $\mathcal{S}$-indexed vector of constants α such that

$$W(s) = W'(s) + \alpha(s) e_K.$$

If $\lambda > 0$, S consists of a single point W^C which is the fixed point of the process. Moreover, we can bound the error of this fixed point with respect to the true log-ratio distribution W^μ by

$$\ell^2_{d_\mu,\lambda}(W^C, W^\mu) \le \frac{1}{1-\hat\gamma}\ell^2_{d_\mu,\lambda}(\Pi_{d_\mu,\lambda,\Phi} W^\mu, W^\mu) - \frac{\hat\gamma\lambda}{1-\hat\gamma}\|W^C - W^\mu\|_{d_\mu, e_K e_K^\top},$$

where the second terms measures the difference in mass between $\hat W$ and W^μ.

We highlight the previous Theorem 4.8, as it provides us with a convergence result independent of the value of $\hat\gamma$ in the distributional setting with Linear Function Approximation; if we recall Theorem 3.5 in the value-based setting, convergence was conditioned to a sufficiently small value of the discounting $\hat\gamma$.

Remark. Hence, we note that our log-distributional approach improves the theoretical convergence guarantees of the process of learning the true ratio distributions. We can perform the learning process in log space, where we can find the additive fixed point W^C, and get our estimate X^C of the true ratio distribution as simply

$$X^C := \exp(W^C)$$

5 Implementation

In the previous sections, the theoretical framework has been presented and analyzed in detail. Our goal now is to describe how we have implemented that framework in practice.

First of all, we highlight the use of Dopamine[34], a research framework for fast prototyping of reinforcement learning algorithms, to implement our Distributional Covariate Shift approach; all the development can be found at our Github repository. Basically, our baseline is the `RainbowAgent`[18] already implemented in Dopamine with the C51 configuration file. The resulting `CovariateShiftAgent` is contained in two files:

- The agent class in `dopamine/agents/covariate_shift/covariate_shift_agent.py`, inheriting from `rainbow_agent.RainbowAgent`.
- The replay buffer in `dopamine/replay_memory/cs_replay_buffer.py`, inheriting from `prioritized_replay_buffer.WrappedPrioritizedReplayBuffer`.

In addition, the configuration file

`dopamine/agents/covariate_shift/configs/covariate_shift.gin`

allows us to easily control all hyper-parameter values of the agent. For more technical information about the implementation, we refer to the API Documentation of our repository, in which we detail each module, class and function that conforms this agent.

In the following subsections, we will analyze the key points of our practical framework at a higher level of abstraction. First, we will describe the function approximation that we use to get our distribution estimates. In the following section, we give account of the sampling process and the replay memory where we store them. Thirdly, we detail how we dealt with the different approximations associated to distributional settings; this facilitates us to finally present in the last two subsections the Categorical Distributional and Categorical Log-Distributional DCOP-TD algorithms, respectively, that we have implemented in our `CovariateShiftAgent`.

5.1 Non-linear Function Approximation

In our attempt to achieve the best experimental performance, we go beyond the defined theoretical framework by implementing a non-linear function approximation for getting ratio -or log-ratio- distribution estimates. This is a choice that prevents us from having any convergence guarantee, but it is supported by the outstanding experimental behaviour of C51[17], which makes use of a non-linear neural network. In fact, in our implementation we augment the original C51 network by adding an extra head, the distributional ratio model, to the final convolutional layer, whose role is to predict the corresponding (log-)ratio distribution.

We can see in Figure 1 the model network architecture in detail when working with the ALE environment[33]. Game frames are selected, grouped and preprocessed in the standard way, generating inputs which consist of a $84 \times 84 \times 4$ image. The following convolutional layers are exactly as the ones already defined in the DQN implementation[5]:

- The first hidden layer convolves 32 filters of 8×8 with stride 4 with the input image and applies a rectifier non-linearity -in particular, a ReLu[35].

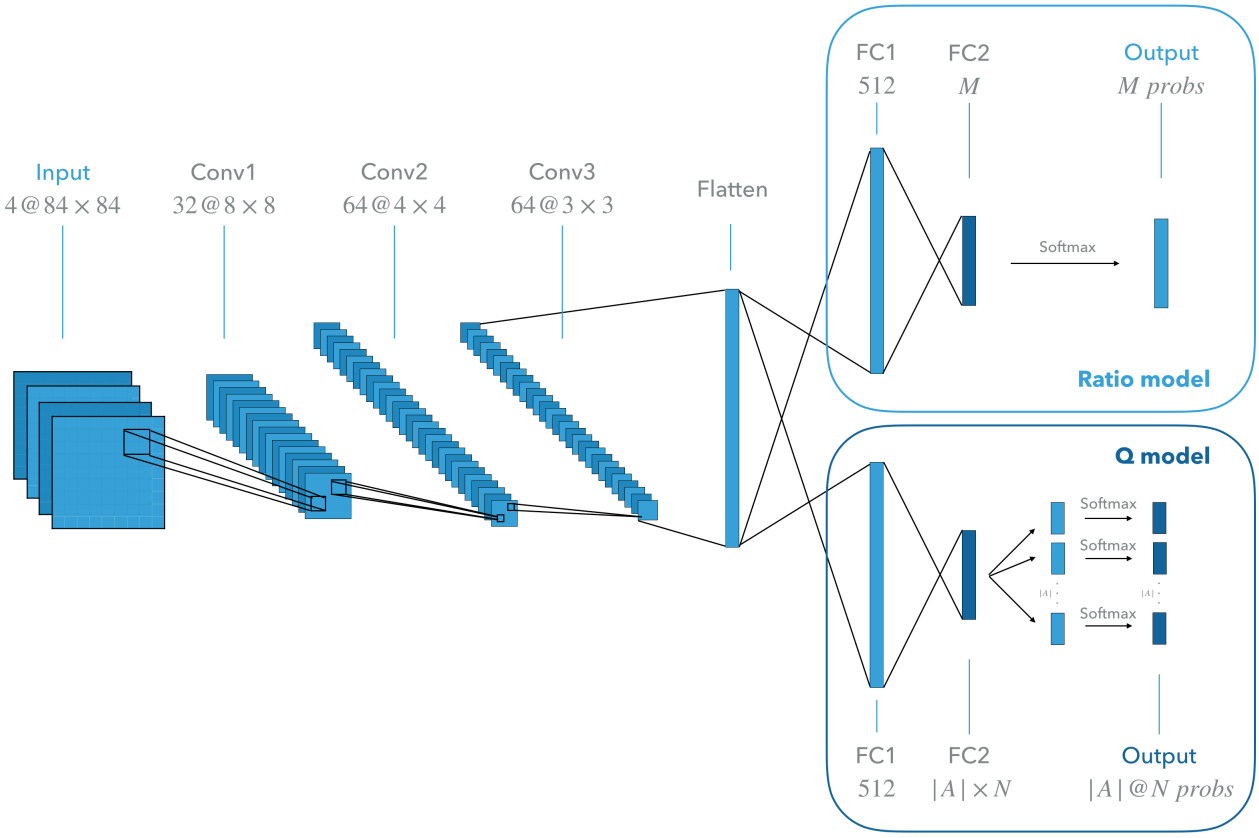

Figure 1: Visual representation of the Neural Network implemented in `CovariateShiftAgent` working in the ALE environment.

- The second hidden layer convolves 64 filters of 4×4 with stride 2, again followed by a ReLu activation function.
- The third and last convolutional layer convolves 64 filters of 3×3 with stride 1 and applies a ReLu rectifier as well, and the result is flattened.

At that point, our network is forked in two separate heads, each connected to the previous flattened layer:

- On the one hand, we have the part responsible of predicting the Q value distributions, which we call the Q model. Just as the original distributional RL models[17], the final hidden layer is fully-connected and consists of 512 rectifier units, followed by a fully-connected linear layer with $|\mathcal{A}| \times N$ outputs, being $|A|$ the number of actions of the played game[5] and N the number of fixed supports considered for the categorical Q distributions. This is re-arranged in $|A|$ groups of N outputs, and applying to each of these groups of logits a Softmax layer we finally obtain the N probabilities for each state-action pair.
- On the other hand, we have added the previously mentioned ratio model, which provides us with estimates of the covariate shift (log-)ratio distributions. As in the Q model, the final hidden layer is fully-connected and consists of 512 rectifier units. This is followed by a fully-connected linear layer that produces as mani outputs as the number of atoms M of the ratio parametric model. A final softmax layer transforms the resulting logits into probabilities.

[5]The number of valid actions varied between 4 and 18 along the games of ALE environment.

Finally, we notice that we work with double networks[11] -i.e. the processes of selection and evaluation of the bootstrap action are decoupled, with online and target weights respectively; this is a usual method in Deep Reinforcement Learning that addresses a natural overestimation bias of Q-learning[36]. In our implementation, the update period for the target network is controlled by `target_update_period` parameter, whose value is set to 8000 iterations just like the `RainbowAgent`[18].

5.2 Replay Memory

The implemented replay memory in Dopamine[34] is responsible of storing past transitions of the agent, acting as a windowed buffer from which training samples are drawn continuously. Consequently, there is always some degree of off-polyceness during the learning process given that the learned policy is being constantly updated.

In order to evaluate our model, however, we are interested in achieving a degree of off-polyceness as high as possible; thus, we consider the very hard setting proposed for testing the Discounted COP-TD algorithm[29], in which:

- We have a fixed behaviour policy μ, the uniformly random policy;
- At each step, the target policy π is the ϵ-greedy policy with respect to the predicted Q estimates.

On ALE[33], this particular setup guarantees that the generated data is significantly different from any learned policy, as we want to.

Taking the work of [29] as reference, to reweight sample transitions we have implemented a prioritized replay memory[28] where priorities are simply covariate shift estimates. Nonetheless, this increases the risk of overfitting, since it reduces the effective size of the data set -those samples unlikely under policy π, which might be the vast majority due to the randomness of μ, end up mostly ignored; to compensate this effect, the `replay_capacity` is increased to 10M frames.

Authors of [29] also explain that stability issues arise when learning the ratio with prioritized sampling. To overcome this, at each training step two independent transition batches are sampled from the replay buffer: prioritized for the value distribution, and uniform for the covariate shift ones. In both cases, the `batch_size` is set to 32 in our experiments.

Last, but not least, we note that initial states cannot be updated through transition samples as the training of the ratio distributions is done 'backwards'. In these cases, given that the distribution of any initial state is policy-independent, their ratio is 1; in our implementation, we take this into account by replacing the bootstrapping target with a Dirac delta centered at 1.

5.3 Distributional Setting

We recall Section 2.3, where we described the approximation framework defined in [25] to implement reproducible and scalable distributional-based RL algorithms. Besides the already detailed use of a non-linear function approximation, it follows how we have dealt in practice with each of the other key points of that framework:

Distribution Parametrisation

Just as we considered along the theoretical framework, we select the parametric family of categorical distributions over some fixed set of supports $z_1 < \cdots < z_K$:

$$\mathcal{P} = \left\{ \sum_{i=1}^{K} p_i \delta_{z_i} \middle| p_1, \ldots, p_K \geq 0, \sum_{k=1}^{K} p_k = 1 \right\}$$

The only difference is that our implementation allows to define supports which are not necessarily equally-spaced; in fact, apart from linear divisions of the support, in our experiments we have also taken into account exponential bins.

Stochastic Learning

Despite the convergence results have been obtained without stochasticity, in practice it is difficult to find an example where it can be rejected. Hence, supported with the work done in [25, 26] regarding stochastic updates, our implementation learns to predict covariate shift ratio estimates through transition samples of the MDP.

In particular, the practical samples for the ratio model are simply of the form (s_t, a_t, s_{t+1}); neither the immediate reward r_t nor the next action a_{t+1} are required. However, in order to define the learning rule, the ratio of policies $\rho(S^{\mu}_{s_{t+1}}, A^{\mu}_{s_t, s_{t+1}})$ must be computed; in our stochastic setting, given a transition (s_t, a_t, s_{t+1}), it is estimated as $\rho_t := \rho(s_t, a_t)$. Since

- the *behavioural policy* μ is simply the uniformly random policy, i.e.

$$\mu(a_t|s_t) = \frac{1}{|\mathcal{A}|} \quad \forall a \in \mathcal{A};$$

- the *target policy* π is the ϵ-greedy policy with respect to the estimated state-action q-values of the model, i.e.

$$\pi_\theta(a_t|s_t) = \begin{cases} (1-\epsilon) + \epsilon \frac{1}{|\mathcal{A}|} & \text{if} \quad a_t = \arg\max_a Q_\theta(s_t, a) \\ \epsilon \frac{1}{|\mathcal{A}|} & \text{otherwise} \end{cases}$$

we can easily obtain a computable expression:

$$\rho_t = \frac{\pi_\theta(a_t|s_t)}{\mu(a_t|s_t)} = \begin{cases} |\mathcal{A}|(1-\epsilon) + \epsilon & \text{if} \quad a_t = \arg\max_a Q_\theta(s_t, a) \\ \epsilon & \text{otherwise} \end{cases} \tag{5.1}$$

Once we have the ratio computed for each sample, it is straightforward to define both the *stochastic Distributional DCOP operator* $\hat{Y}^D_{\hat{\gamma}}$ and the *stochastic Log-Distributional DCOP operator* $\hat{G}_{\hat{\gamma}}$, which modify the supports of ratio distributions and log ratio distributions, respectively, according to the 1-step information contained in each sample.

Projection of Bellman Target Distribution

In order to recover a ratio distribution within $\mathcal{P}$ after applying the stochastic operator -either $\hat{Y}^D_{\hat{\gamma}}$ or $\hat{G}_{\hat{\gamma}}$, depending on the case-, we also use the heuristic projection operator Π_C defined in 2.16; in particular, we coded it extended to finite mixtures of Dirac measures:

$$\Pi_C \left(\sum_{i=1}^{N} p_i \delta_{y_i} \right) = \sum_{i=1}^{N} p_i \Pi_C(\delta_{y_i})$$

We note, however, that our implementation of the function `project_distribution(...)`[6], responsible of performing this projection Π_C, differs from the already implemented in Dopamine; we have slightly modified it so as to allow non-equally spaced supports, thus enabling the option of running experiments with exponential bins as well.

Gradient Updates

Attending to what we have shown in our theoretical framework, defining a gradient update based on a Cramér loss seem to be the most appropriate solution given that we have implemented the Cramér projection Π_C. However, we finally decided to imitate the original C51 agent and perform a single step of gradient descent on the Kullback-Leibler divergence of the predicted distribution from the target one with respect to the parameters of the prediction.

The reasons behind our decision are purely practical; despite there is no convergence guarantee of C51 due to this KL divergence step, so far its experimental results[17] are much better than those that strictly rely on the theoretical framework[26]. Hence, as we now seek for the best performance possible of our Distributional Covariate Shift approach, we kind of ignore that part of the theory and implement the KL gradient update for generating the new estimates of our `CovariateShiftAgent`.

5.4 Categorical Distributional DCOP-TD

With all the main ingredients of our implementation detailed in previous sections, next we summarize the steps that conform our distributional DCOP-TD algorithms. To begin with, we show in Algorithm 2 the Categorical Distributional $\hat{\gamma}$-Discounted COP-TD main process, where

- $\{x_K\}$ is the set of atoms that define our categorical parametric family $\mathcal{P}$
- we denote by θ and $\bar{\theta}$ the weights of the online and target networks, respectively;
- $\{p_{\theta_t,K}(s)\}$ and $\{p_{\bar{\theta}_t,K}(s)\}$ are, respectively, the set of probability outputs of the online and target networks for a state $s \in \mathcal{S}$ at a training time step t.
- $\hat{\epsilon} \in [0,1]$ comes from the definition of the ϵ-greedy policy $\pi_{\hat{\theta}}$; in practice, however, it can be controlled apart by an extra hyper-parameter (`quotient_epsilon`).
- $(f_{a,b})_{\#}$ represents the application to each element of the support of an affine shift map $f_{a,b} : \mathbb{R} \to \mathbb{R}$ defined by $f_{a,b}(y) := a+by$; in particular, note that $Y^D_{\hat{\gamma}} X(s) := (f_{\cdot,\hat{\gamma}\rho})_{\#} X(s)$

[6]This function can be found in dopamine/agents/covariate_shift/covariate_shift_agent.py

Algorithm 2: Categorical Distributional DCOP-TD

Require: Estimates $X_{\theta_t}(s) = \sum_{k=1}^{K} p_{\theta_t,k}(s)\delta_{x_k}$ and $X_{\bar{\theta}_t}(s) = \sum_{k=1}^{K} p_{\bar{\theta}_t,k}(s)\delta_{x_k}$ for each $s \in \mathcal{S}$

Input: A sample transition (s_i, a_i, s_{i+1})

`#Compute policy ratio`

if $a_t = \arg\max_a Q_{\bar{\theta}}(s_i, a)$ **then**

$\rho_i \leftarrow |\mathcal{A}|(1-\hat{\epsilon}) + \hat{\epsilon}$

else

$\rho_i \leftarrow \hat{\epsilon}/|\mathcal{A}|$

end if

`#Compute distributional ratio target`

$\widehat{X}_*(s_{i+1}) \leftarrow Y^D_{\hat{\gamma}} X_{\bar{\theta}_t}(s_i)$

`#Project target onto support`

$\widehat{X}_t(s_{i+1}) \leftarrow \Pi_C \widehat{X}_*(s_{i+1})$

`#Compute KL loss`

Find gradient KL $\left(\widehat{X}_t(s_{i+1}) || X_{\theta_t}(s_{i+1})\right)$

Update online weights θ_{t+1}

`#Update target network if required`

$\bar{\theta}_{t+1} \leftarrow \theta_{t+1}$ **if** t % `target_update_period` $= 0$ **else** $\bar{\theta}_t$

Output: New estimates $X_{\theta_{t+1}}(s) = \sum_{k=1}^{K} p_{\theta_{t+1},k}(s)\delta_{x_k}$ and $X_{\bar{\theta}_{t+1}}(s) = \sum_{k=1}^{K} p_{\bar{\theta}_{t+1},k}(s)\delta_{x_k}$ for each $s \in \mathcal{S}$

Ratio estimates $c_{\bar{\theta}_{t+1}}(s) = \sum_{k=1}^{K} p_{\bar{\theta}_{t+1},k}(s)x_k$

5.5 Categorical Log-Distributional DCOP-TD

Finally, we present in Algorithm 3 the sketch of our Categorical Log-Distributional DCOP-TD algorithm. The observations made in previous Section apply here, where now $\{w_K\}$ is the support and $G_{\hat{\gamma}}W(s) = (f_{\log(\rho),\hat{\gamma}})_{\#}W(s)$. In fact, as one can see, the steps are equivalent to those of the multiplicative case.

Algorithm 3: Categorical Log-Distributional DCOP-TD

Require: Estimates $W_{\theta_t}(s) = \sum_{k=1}^{K} p_{\theta_t,k}(s)\delta_{w_k}$ and $W_{\bar{\theta}_t}(s) = \sum_{k=1}^{K} p_{\bar{\theta}_t,k}(s)\delta_{w_k}$ for each $s \in \mathcal{S}$

Input: A sample transition (s_i, a_i, s_{i+1})

`#Compute the log policy ratio`

if $a_t = \arg\max_a Q_{\bar{\theta}}(s_i, a)$ **then**

$\log(\rho_i) \leftarrow \log\left(|\mathcal{A}|(1-\hat{\epsilon}) + \hat{\epsilon}\right)$

else

$\log(\rho_i) \leftarrow (\hat{\epsilon}/|\mathcal{A}|)$

end if

`#Compute distributional log-ratio target`

$\widehat{W}_*(s_{i+1}) \leftarrow (f_{\log(\rho_i),\hat{\gamma}})_{\#}W_{\bar{\theta}_t}(s_i)$

`#Project target onto support`

$\widehat{W}_t(s_{i+1}) \leftarrow \Pi_C \widehat{W}_*(s_{i+1})$

`#Compute KL loss`

Find gradient KL $\left(\widehat{W}_t(s_{i+1}) || W_{\theta_t}(s_{i+1})\right)$

Update online weights θ_{t+1}

`#Update target network if required`

$\bar{\theta}_{t+1} \leftarrow \theta_{t+1}$ **if** t % `target_update_period` $= 0$ **else** $\bar{\theta}_t$

Output: New estimates $W_{\theta_{t+1}}(s) = \sum_{k=1}^{K} p_{\theta_{t+1},k}(s)\delta_{w_k}$ and $W_{\bar{\theta}_{t+1}}(s) = \sum_{k=1}^{K} p_{\bar{\theta}_{t+1},k}(s)\delta_{w_k}$ for each $s \in \mathcal{S}$

Ratio estimates $c_{\bar{\theta}_{t+1}}(s) = \sum_{k=1}^{K} p_{\bar{\theta}_{t+1},k}(s)\exp(w_k)$

We note that we can easily get covariate shift estimates -which we recall are used as priorities for training the Q model- by exponentiating log-ratio distributions and taking its mean; in practice, as the support is fixed, we simply compute the exponentiated support at the beginning, and compute its weighted mean by the corresponding probabilities of log-ratio distributions.

6 Evaluation of the Proposal

In this section we aim to evaluate our implementation by showing some experimental results within ALE[33] which we recall it is an interface to Atari 2600 games; a detailed explanation of this platform can be found in Appendix A.4. In order to help us with the analysis, we take as reference the results achieved in [29] with the value-based DCOP-TD algorithm.

However, before presenting and detailing the experiments performed, we comment on the practical limitations that we have faced when testing these RL algorithms. As one might suspect, everything is about the associated computational cost: a single run of 100 iterations on any game requires +5 days with a constant use of +60GB of RAM (due to the large replay buffer) and a dedicated GPU[7].

We were therefore constrained by the available resources when planning the set of experiments. In particular, this prevented us from designing a systematic grid search so as to select the hyper-parameter values of our distributional approach; instead, these values were selected either by reusing the published values of previous works[17, 29], or by performing an informal search on the game of Seaquest. We summarize in the following list the most relevant selections:

- The number of atoms of the ratio parametric family $\mathcal{P}$ (`ratio_num_atoms`) is set to 51, as in C51[17].
- The covariate shift interval definition of $\mathcal{P}$ is defined so that it is exponentially symmetric with respect to the maximum and minimum possible values of the policies quotient ρ_t; recalling its practical definition in Equation 5.1, we have
  - `ratio_cmin` $= \epsilon^a$
  - `ratio_cmin` $= (|\mathcal{A}|(1-\epsilon)+\epsilon)^a$

  being ϵ the effective `quotient_epsilon` and a the selected `symmetric_exponent` (in our experiments, $a = 2$). Despite originally the interval was manually defined, we found that enabling this kind of `symmetric_range` dramatically improved the performance[8].
- The mentioned effective `quotient_epsilon`, which is used to compute the policy quotients, is set to 0.5; theoretically, it should be equal to the `epsilon_eval` used for defining the ϵ-greedy policy in the evaluation setting (which is 0.1), but in practice incrementing it -i.e. reducing the variance of the updates- provided us with better results.
- We use a discount factor $\hat{\gamma}$ (`ratio_discount_factor`) of 0.97; in our tests, though, this parameter did not seem to be critical.
- The weight loss of the ratio model (`ratio_weight_loss`) is 0.002, as in [29].

Bearing that practical setting in mind, in this section we first compare the experimental results of both the multiplicative and the logarithmic versions of our distributional approach on a single game (Seaquest). Then, we extend the analysis to three more games in the case of the Categorical Log-Distributional DCOP-TD. Thirdly, we perform a qualitative evaluation of the learned ratios at the end of the previous experiments. Finally, we provide some evidence that the resulting covariate shift estimates actually represent an 'Off-policyness' measure.

[7] In our case, a Nvidia GTX 1080 Ti.

[8] The idea is that, starting from a Dirac delta function centered at 1, the same number of max. and min. updates can be applied, respectively, so that the resulting target distributions still lie in the interval.

6.1 Multiplicative vs. Additive Value Functions

Our first goal was to compare the Categorical Distributional DCOP-TD with the multiplicative value function with the Categorical Log-Distributional version that exhibits an additive one. Even though our actual implementation of both versions lack of theoretical convergence guarantees, we recall that the Logarithmic approach showed a theoretically stronger convergence behaviour with linear function approximation and the Crámer-based loss; we wanted to see if this might translate into better performance in our practical setting.

In addition to that, we were also interested in comparing the results with those obtained with the value-based DCOP-TD algorithm presented in [29], as well as with the performance obtained when no covariate shift estimates are used as priorities for the Q model- we will refer to it as the uncorrected case following the notation of [29].

Due to the previously commented hardware limitations, though, we restricted the analysis to single game -Seaquest- with a maximum of 100 iterations; Figure 2 shows the attained learning curves of the considered models.

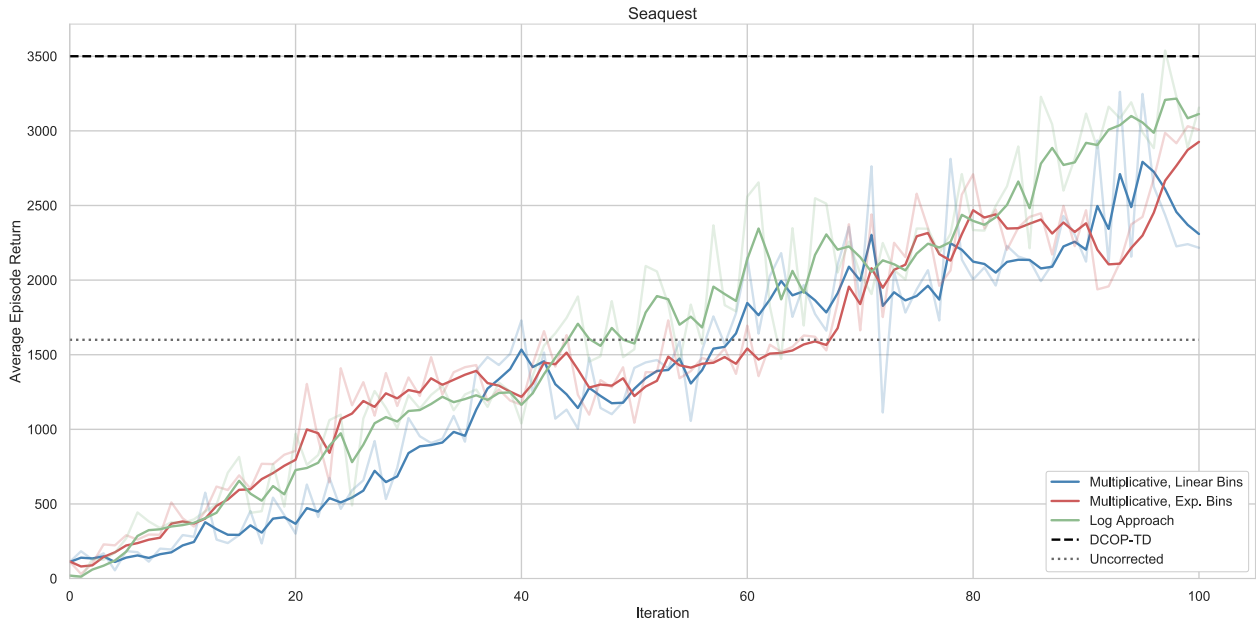

Figure 2: Performance comparison of Categorical Distributional DCOP-TD -with both linear and exponential bins- and its Categorical Log-Distributional counterpart. Trajectories have been smoothed by applying an Exponential Moving Average, and straight lines represent performances of the value-based DCOP-TD algorithm and its uncorrected version at 100 iterations, both extracted from [29].

As we can see in that Figure 2, similar performances are achieved by both versions; the logarithmic approach seems to get a better trend at 100 iterations, but we do not find the difference with respect the multiplicative case to be significant. Regarding the use of linear or exponential bins in the Categorical Distributional DCOP-TD, there is no evidence that one way behaves better than the other either.

However, it is interesting to note that all distributional-based algorithms clearly overpass the uncorrected case, and that performances at 100 iterations are comparable -especially the Log-Distributional Approach- to the DCOP-TD of [29], where the latter was exhaustively fine-tuned. These results suggest that the ratios learned by our distributional models actually help in the learning process.

6.2 Log-Distributional Approach

Having observed the experimental results of the different distributional configurations on the game of Seaquest, next we wanted to extend the study to other Atari2600 games. For doing so, we consider the Categorical Log-Distributional DCOP-TD, our most innovative approach with an associated additive value function.

In order to perform a comparison with the value-based DCOP-TD, we have selected three more games -besides Seaquest- that were also analyzed in [29]: Breakout, Asterix and Pong. Figure 3 show the resulting learning curves.

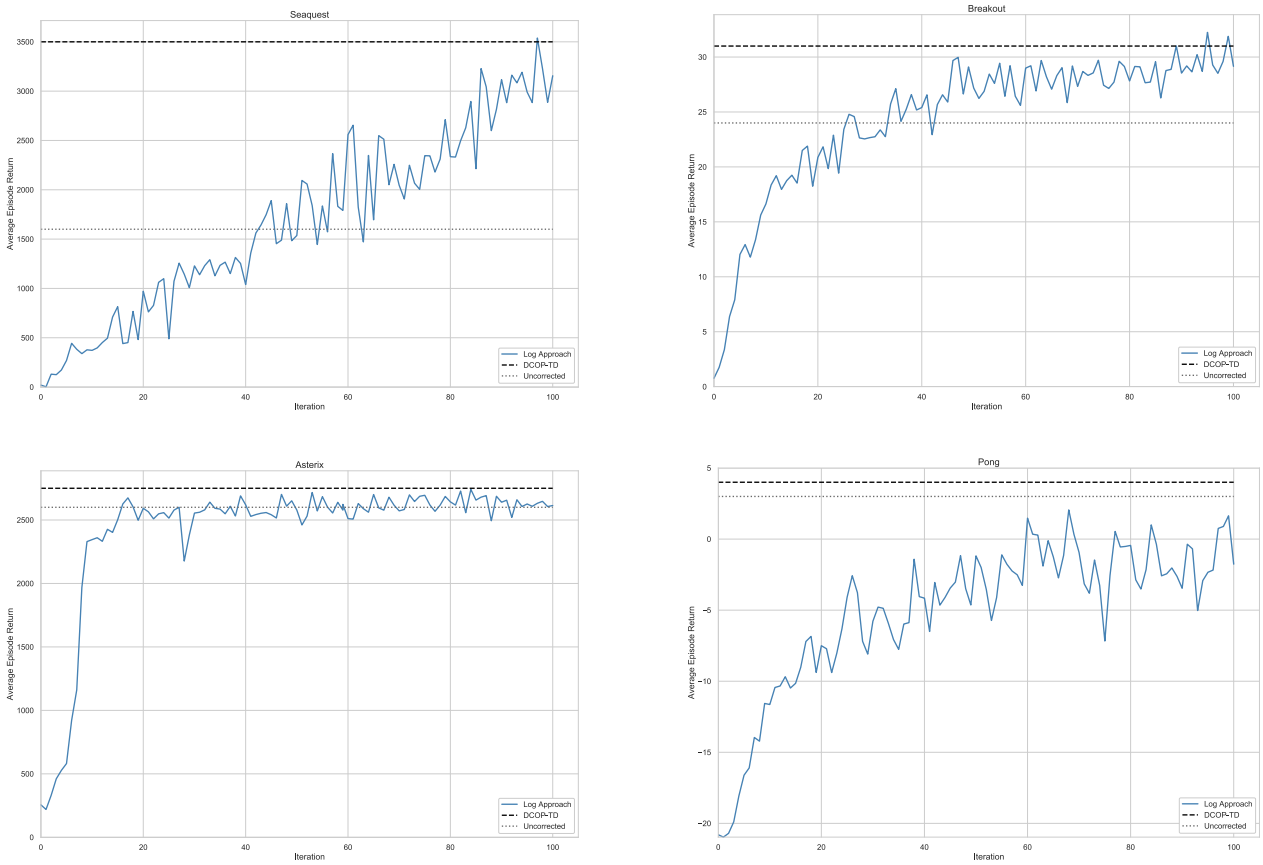

Figure 3: Performance of Categorical Log-Distributional DCOP-TD on four different games within ALE. Straight lines represent performances of the value-based DCOP-TD algorithm and its uncorrected version at 100 iterations, both extracted from [29].

As one can see, at 100 iterations our implementation attains a performance in Breakout very similar to that of DCOP-TD in [29], and clearly outperforms the uncorrected case -just like we noted on Seaquest. The results on Asterix are more difficult to interpret, as the learning curve seems to lie just between the DCOP-TD and the uncorrected references, which in this case are quite close to each other.

However, what is also evident is that our Log-Approach obtained poor results on the game of Pong, where even the uncorrected case achieved a better performance. We suspect that this behaviour might be improved by doing a more careful selection of the hyper-parameter configuration.

In general, though, the results still suggest that the use of our log-distributional approach is potentially beneficial to the learning process when dealing with off-policy data.

6.3 Qualitative Evaluation of the Learned Ratios

In addition to the previous experiments, we were interested in qualitatively assessing the learned covariate shift estimates, and for doing so we followed the experimental design provided in [29]: we take 100.000 samples drawn from the random behaviour policy on a certain game, and select among them the top and bottom states according to the ratio (c) predicted by the previously trained agents.

We recall that a covariate shift estimate greater than 1 means that the network believes the state is more likely under the target policy π than under the behaviour policy μ, whereas $c < 1$ is the other way around. Hence, given that π results in a much more successful policy than μ at the considered stages of the learning process, we should expect to find larger ratios associated with good moves, and lower ratios to bad ones.

So as to facilitate the reasoning of how good or bad are the selected frames, we present the results on the game of Breakout; not only it is easily interpretable, but moreover, as we showed in the previous section, on this game our trained Log-Distributional agent attained a very good performance. Hence, Figures 4 and 5 show the top 16 frames with greater and lower predicted ratio, respectively.

Qualitatively speaking, one can observe that frames with highest covariate shift estimates correspond to better contexts than those with the lowest, as the agent is either about to return the ball or failing to give it back for little. In contrast to that, frames with the lowest c predictions match clearly with situations where the ball has not been returned back; in fact, in most cases we can not even see the ball, meaning that the 'goal' had been already scored.

Therefore, by observing this clear trend we conclude that our distributional model is actually learning to distinguish between likely and unlikely states under policies π and μ, just as we would like to. As it was stated in [29], this might be particularly significant given the relative scarcity of off-policy methods of this kind in deep reinforcement learning.

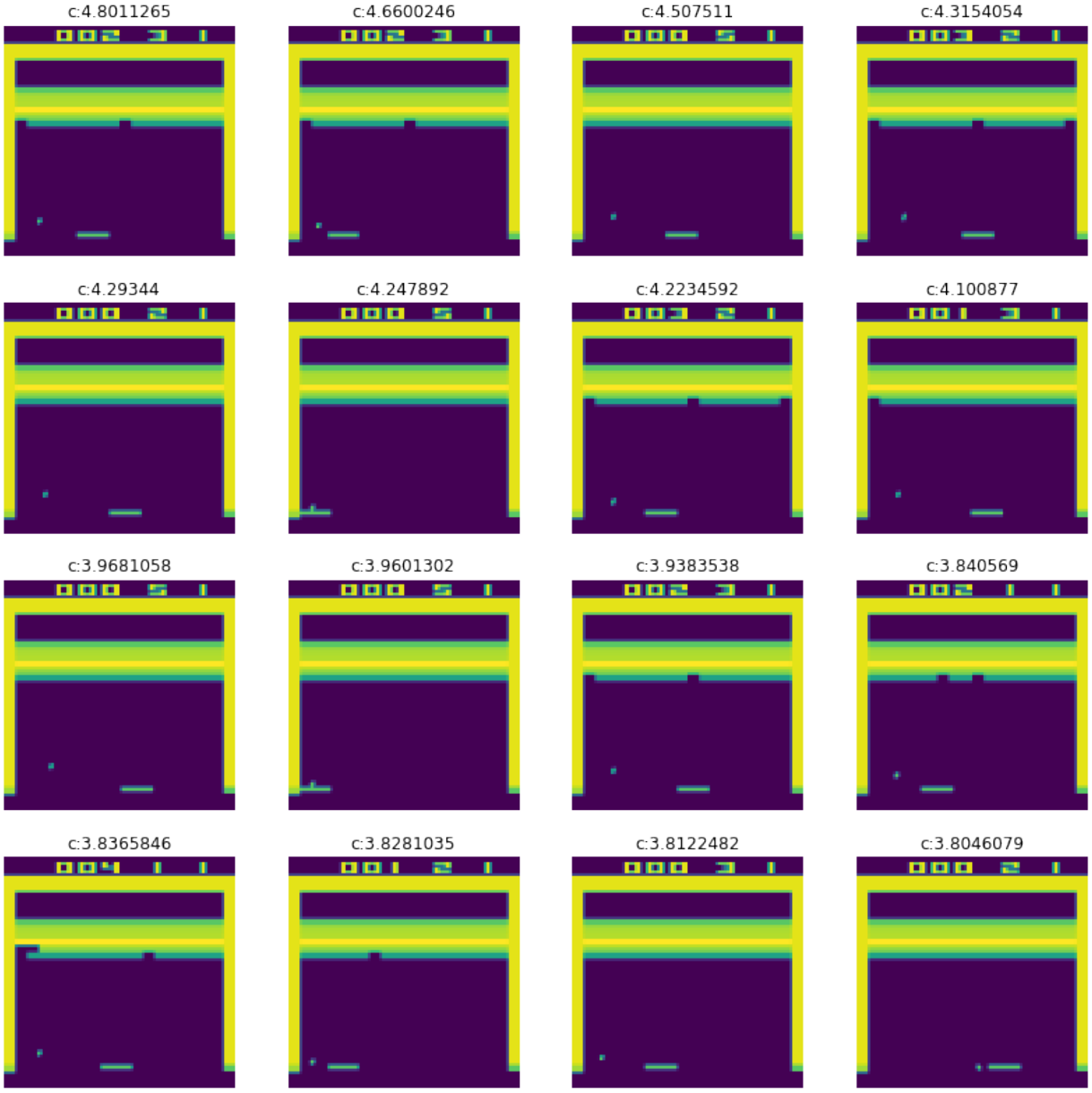

Figure 4: Breakout frames with the highest c value predictions among the 100.000 samples generated by the random behaviour policy μ.

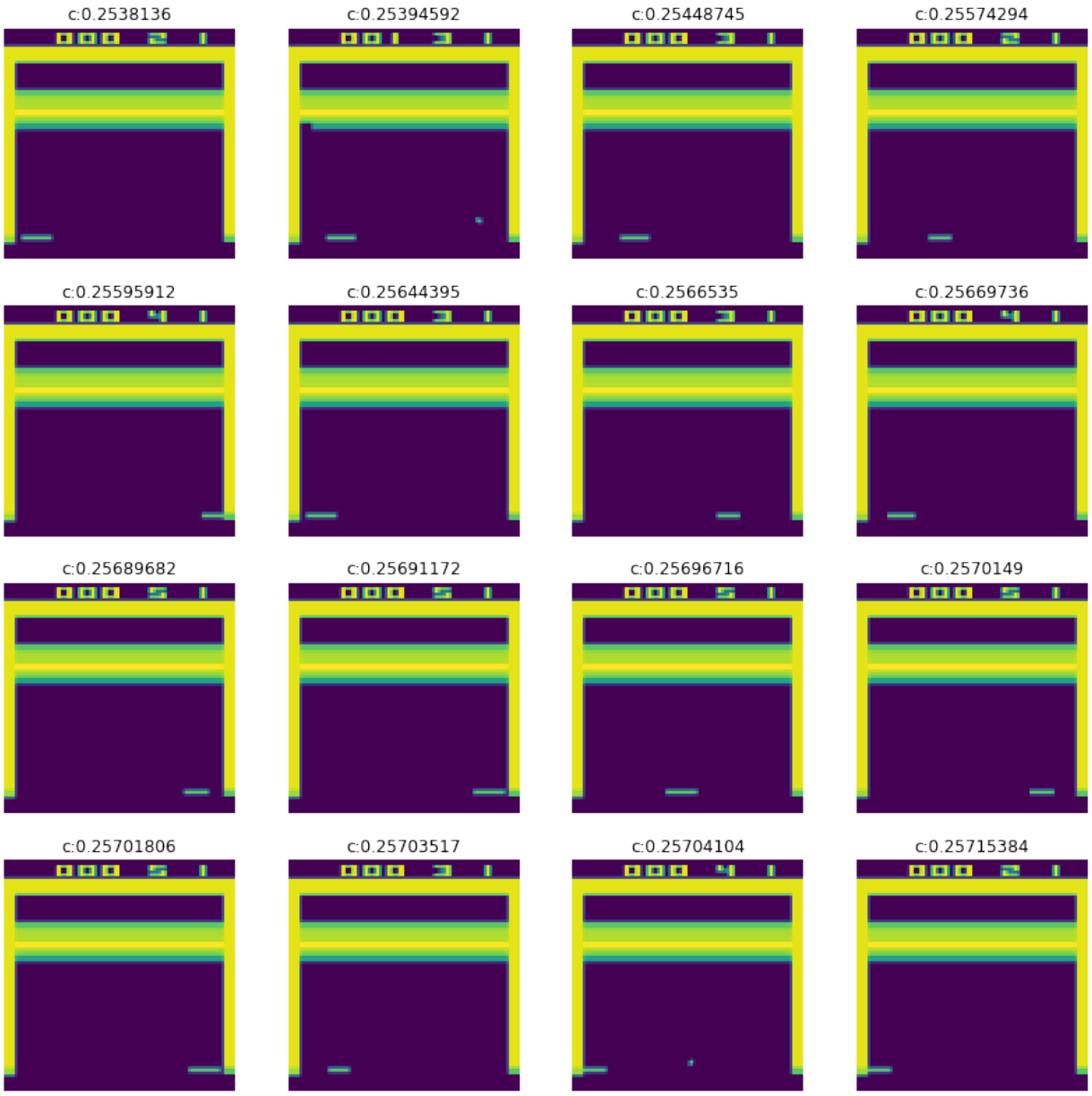

Figure 5: Breakout frames with the lowest c value predictions among the 100.000 samples generated by the random behaviour policy μ.

6.4 Measuring Off-policy Degree

So far, in all experiments we have considered the extreme case of a fixed random behaviour policy μ, which clearly generates constantly great amounts of off-policy data with respect to the target policy π. However, in more general situations we might find behaviour policies that are periodically updated, and whose strategies to select an action may be much similar to those of the target policy.

With this idea in mind, we finally designed an additional experiment to check whether the learned ratios of our model are also sensitive to different degrees of Off-policy data, as they should be. For doing so, we again use a previously trained Log-Distributional agent, in this case on the game of Seaquest. The setup is summarized as follows:

- We only train the ratio head of the agent deep network (recall Figure 1), considering a fixed Q model. In particular, note that this implies fixing the target policy π, defined as an ϵ-greedy policy (with $\epsilon = 0.1$) with respect to the maximum Q value predictions.
- We reset the replay buffer, and for the first million training frames we set $\mu = \pi$, so the agent only sees on-policy data.
- At 1M frames, we switch to a behaviour policy which is able to generate some amount of off-policy data; for that, we also define μ as an ϵ-greedy policy with respect to the Q predictions, but with $\epsilon > 0.1$.
- The ratio model is trained for another 5M frames, and during this process more and more off-policy samples are stored in the replay buffer.

Note that, in this setting, we are facing for the first time a multiple policy data problem; in order to deal with this, we needed to store the probability of taking each action of the agent trajectory (i.e $\mu(a_t|s_t)$ of each sample (s_t, a_t, s_{t+1})) to then compute the policy quotient ρ_t at each update in a proper way.

One way of evaluating the results of this experiment would be following the evolution of the predicted ratio values along the whole process, but to fully capture what might be happening to them we would probably need to keep track of their maximum, minimum and mean values. Instead, we came up with another measurement that might directly quantify the off-policy degree from the ratio estimates.

It is straightforward to argue that a certain covariate shift estimate $c = k$, with $k > 1$, indicates that its associated state is as off-policy as another state whose ratio is $1/k$. Bearing that in mind, we define the 'Off-policyness' measure of a state as its covariate shift if it is greater or equal than 1, or as the reciprocal of that ratio otherwise. We strongly believe that keeping track of this single new magnitude would be enough to fully understand the results of the described experiment.

This is precisely what it is shown in Figure 6, where the batch average 'Off-policyness' is represented along four different experiments, each one switching to a different ϵ-greedy behaviour policy μ at 1M frames. In particular, note that with $\epsilon = 1$ we are actually considering the random policy.

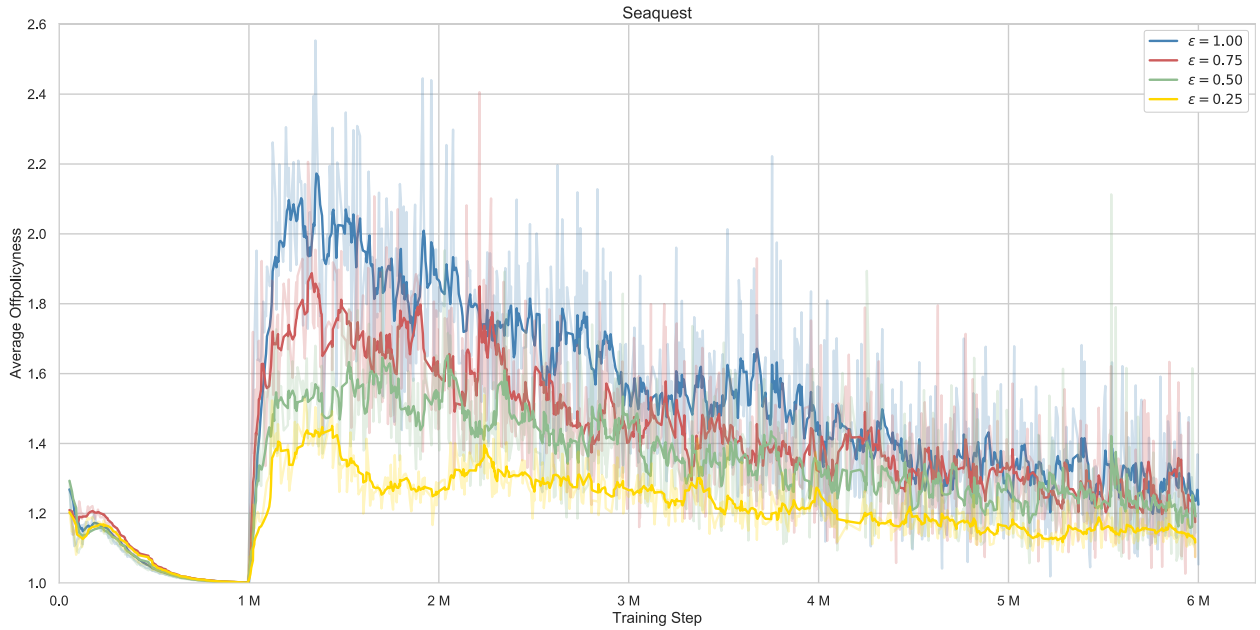

Figure 6: Resulting 'Off-Policyness' curves considering 4 different ϵ-greedy behaviour policies to switch at 1M frames. Trajectories have been smoothed by applying an Exponential Moving Average.

We first want to emphasize the observed behaviour during the first million frames: as one should expect, we rapidly end up with an 'Off-policyness' convergence to 1, and hence covariate shift ratio estimations of 1, which simply means that we are dealing exclusively with on-policy data.

Then, starting at 1M frames, it is interesting how the gap on 'Off-policyness', produced by the switch of the behaviour policy, is related to ϵ values: the higher the ϵ considered, the steeper the gap obtained. In fact, we argue this is also the desired behaviour: as we increase ϵ, the random component of μ arises, which makes its action selection strategy to differ more and more from the that of the target policy π, so the generated samples are potentially more off-policy.

After the abrupt gap, one can see how the average 'Off-policyness' slowly decreases in all four cases. Since we are only training the ratio model, and the expected policy quotient is 1 even with off-policy data, this mild convergence to 1 could be also hoped.

To sum up, we conclude that the results of this experiment provide some empirical evidence that the learned ratios of our model are able to measure, to some extent, the off-policy degree of the data that we are using.

7 Conclusions

We have revisited some fundamental theoretical results of Distributional Reinforcement Learning[17] when using a categorical parametrization for return distributions[25, 26]. This review, which brings forth the connection between distributional operators and mixture distributions, includes a detailed analysis of the convergence guarantees of distributional Bellman updates when combined with linear function approximation.

On the other hand, we presented a thorough inspection of the Covariate Shift method[27, 29] designed to deal with Off-Policy Learning[1]. As we observed, this approach proposed a multiplicative TD-based update rule for learning the ratios between the target and behaviour stationary distributions which, together with linear function approximation, provides with provable -but not strong- convergence guarantees.

Motivated by the interesting new properties of distributional-based RL, which derive from the aforementioned connection with mixtures, we introduced a theoretical framework for learning covariate shift ratios from a distributional perspective. Although expected, it was still surprising to find out that such a distributional approach allows to project the ratio learning process to the log space, moving from a multiplicative update rule to an additive one. We can then look for the additive distributional fixed point in that log space, so that exponentiating it leads to the correct covariate shift ratio distribution.

More precisely, we observed that the learning process of log-distributional covariate shift ratios can be actually described by a distributional Bellman operator. In particular, this let us make use of the -already reviewed- theoretical results of Categorical Distributional RL[25, 26] in order to define a Categorical Log-Distributional Covariate Shift approach with better convergence guarantees than previous value-based methods[27, 29].

For assessing this new distributional perspective applied to covariate shift ratios, we implemented two different solutions: an algorithm that learns the ratio distributions using multiplicative updates (Categorical Distributional DCOP-TD), and other that performs the learning in the log space (Categorical Log-Distributional DCOP-TD).

Despite more experiments need to be run, still preliminary results on a subset of Atari 2600 games demonstrates the practicality of the learned covariate shift ratio distributions in an off-policy setting. In practice, we do not observe any significant deviation between learning with an additive or a multiplicative value function, suggesting that differences in their convergence behaviour might be mainly a theoretical discussion. When comparing our distributional approach to the value-based DCOP-TD[29], our reference in the extreme setting we are dealing with, similar performances are attained even with the limited fine-tuning of our model. In addition, we provide an empirical proof that the learned covariate shift estimates are sensitive to the off-policy degree of the data seen by the agent.

Future Work

Our developed theoretical framework for designing a distributional covariate shift approach can be easily broaden to any other context where learning is ruled by a non-additive value function, and we believe its use might be specially relevant in those cases that present instability issues. Related to that, it would be interesting to look for some practical case where the theoretical guarantees of the Categorical Log-Distributional Approach actually make a difference with respect to learning from its original multiplicative update rule. Finally, extending our whole implementation to deal with the more general case where the agent might learn from multiple-policy data would also be an exciting avenue.

Appendix

A.1 Contraction Mappings

Contraction Mappings are key for our theoretical framework to be understood, as they allow us to assess the convergence guarantees of the different involved algorithms. In this section, we define what a Contraction Mapping is, as well as present the *Contraction Mapping Theorem* -better known as the *Banach Fixed-point Theorem*-, which is in fact the method that we apply to construct the solution of convergence results.

Let us begin with the definition:

Definition 1. Consider a metric space (X, d). A mapping $T : X \to X$ is a *contraction mapping*, or *contraction*, if there exists a constant $c \in (0, 1)$ such that, for all $x, y \in X$,

$$d\left(T(x), T(y)\right) \leq c \cdot d(x, y) \tag{A.1}$$

In the limit where A.1 holds with $c = 1$, we denote T as a *non-expansion mapping*, or simply *non-expansion*.

The idea, hence, is that contractions perform uniformly continuous mappings of the points of the considered metric space so that they end up closer together.

Notation. To simplify the notation, the parenthesis around the argument of a map T are usually omitted (i.e. we write Tx instead of $T(x)$). Moreover, when applying the map repeated times to the same point $x \in X$, we denote the n-th iterate of T as T^n.

For any mapping $T : X \to X$, we recall that a point $x \in X$ such that

$$Tx = x \tag{A.2}$$

is called a *fixed point* of T. Precisely, the interest of contractions is intimately related to the properties of their fixed points, which are summarized in the mentioned Contraction Mapping Theorem. However, this theorem requires the metric space considered to be *complete*, which in turn involves the notion of *Cauchy sequences*:

Definition 2. A sequence $\{x_n\}$ in a metric space (X, d) is a *Cauchy sequence* if for every $\epsilon > 0$, there exists some $n_0 \in \mathbb{N}$ such that

$$d(x_n, x_m) < \epsilon$$

for all $n, m \geq n_0$.

Definition 3. A metric space (X, d) is *complete* if every Cauchy sequence in it converges.

Now we have all ingredients to present the theorem, widely known also as the *Banach Fixed-point Theorem*, which demonstrates the existence and uniqueness of fixed points of contraction mappings on complete metric spaces. We note that, in general, the condition $c < 1$ is needed for proving that exists the fixed point and it is unique.

Theorem A.1. *Let $T : X \to X$ be a contraction mapping on a complete metric space (X, d). Then, T has exactly one fixed point $x \in X$.*

Proof. The proof explicitly constructs a sequence converging to the fixed point.

Consider any point $x_0 \in X$; we define a sequence $\{x_n\}$ in X by

$$x_{n+1} = Tx_n$$

for $n \geq 0$. In particular, we have $x_n = T^n x_0$.

We first prove that $\{x_n\}$ is a Cauchy sequence. Considering $n \geq m \geq 1$, and the contraction definition A.1, we have

$$\begin{aligned} d(x_n, x_m) &= d(T^n x_o, T^m x_0) \\ &\leq c^m d(T^{n-m} x_0, x_0) \end{aligned}$$

By the triangle inequality, we can follow the expansion as

$$\begin{aligned} d(x_n, x_m) &\leq c^m \sum_{i=0}^{n-m-1} d(T^{n-m-k} x_0, T^{n-m-k-1} x_0) \\ &\leq c^m \left(\sum_{i=0}^{n-m-1} c^k \right) d(Tx_0, x_0) \\ &\leq c^m \left(\sum_{i=0}^{\infty} c^k \right) d(x_1, x_0) \\ &\leq \left(\frac{c^m}{1-c} \right) d(x_1, x_0) \end{aligned}$$

From the previous expression it is straightforward that $\{x_n\}$ is Cauchy, and since the metric space is complete, it converges to a limit point $x \in X$. The fact that this limit x is a fixed point can be easily derived from the continuity of T:

$$Tx = T \lim_{n\to\infty} x_n = \lim_{n\to\infty} Tx_n = \lim_{n\to\infty} x_{n+1} = x$$

Finally, we prove the uniqueness of x. Suppose x and y are two fixed points; hence,

$$0 \leq d(x, y) = d(Tx, Ty) \leq c \cdot d(x, y)$$

But as $c \in (0, 1)$, we have $d(x, y) = 0$, so $x = y$. □

We highlight that the result of Theorem A.1 do not depend on the metric d: for any metric on X that makes X complete and T a contraction on X, the existence and uniqueness of a fixed point is guaranteed.

A.2 Mixture Distributions

In this section we introduce Mixture Distributions, whose concept and properties are relevant for understanding and developing the distributional theoretical framework presented in this document.

Remark. All random variables presented in this section are considered to be real-valued, i.e. their measurable space is $E = \mathbb{R}$.

Definition 4. A random variable Y is a *mixture distribution* if it is derived from a collection of other random variables $\{X_i\}$, $i \in \{1, \dots, N\}$, (named *mixture components*) in such a way that the combination of these parent distributions is driven according to a certain distribution A (called *mixing distribution*). A encapsulates the mixture weights $\alpha_i \sim A$, $i \in \{1, \dots, N\}$, which represent the probabilities of each individual mixture component X_i.

The mixture distribution Y can be defined in terms of its density function f_Y, which is the resulting α-convex combination of the mixture components' density functions:

$$f_Y(x) = \sum_{i=1}^{N} \alpha_i f_{X_i}(x)$$

Mixture Distributions appear in a natural way when dealing with distributional versions of Bellman equations, and are of special interest due to some interesting properties:

Property 1. *The expectation of the mixture distribution Y is the convex combination of expectations of each mixture component:*

$$\begin{aligned} \mathbb{E}[Y] &= \int_{-\infty}^{\infty} x f_Y(x) dx = \int_{-\infty}^{\infty} x \sum_{i=1}^{N} \alpha_i f_{X_i}(x) dx \\ &= \sum_{i=1}^{N} \alpha_i \int_{-\infty}^{\infty} x f_{X_i}(x) dx \\ &= \sum_{i=1}^{N} \alpha_i \, \mathbb{E}[X_i] \end{aligned} \tag{A.3}$$

Property 2. *Let be $Z = g(Y)$, being Y a mixture distribution with mixture components $\{X_i\}$, $i \in \{1, \dots, N\}$, and g a monotonic, invertible and differentiable function. Then*

$$\begin{aligned} f_Z(x) &= f_Y(g^{-1}(x)) \left| \frac{dg^{-1}(x)}{dx} \right| \\ &= \sum_i \alpha_i f_{X_i}(g^{-1}(x)) \left| \frac{dg^{-1}(x)}{dx} \right| \\ &= \sum_i \alpha_i f_{g(X_i)}(x) \end{aligned} \tag{A.4}$$

We emphasize the relevance of **Property 2** in proposing our Categorical Log-Distributional approach (Section 6.2). In distributional TD learning, the distribution mixture plays the role of the expectation in expected TD. But while $\mathbb{E}[g(x)] \neq g(\mathbb{E}[x])$, we can interchange mixtures and functions. This allows us to circumvent Jensen's inequalities.

A.3 Measures over Distributions

In this section we present the definitions of the measures and metrics over distributions that are used at some point of our theoretical framework.

A.3.1 Kullback-Leibler Divergence

Definition 5. Let be $\nu_1, \nu_2 \in \mathscr{P}(\mathbb{R})$ two probability distributions. The Kullback-Leibler (KL) divergence of ν_1 from ν_2 is defined as

$$D_{KL}(\nu_1, \nu_2) = \int_{-\infty}^{\infty} \nu_1(x) \log \frac{\nu_1(x)}{\nu_2(x)} dx$$

A.3.2 Wasserstein

Definition 6. The Wasserstein distance d_p, for $p \in [1, \infty)$, between two distributions $\nu_1, \nu_2 \in \mathscr{P}(\mathbb{R})$, with cumulative distribution functions F_{ν_1}, F_{ν_2} respectively, can be defined by:

$$d_p(\nu_1, \nu_2) = \left(\int_{\mathbb{R}} |F_{\nu_1}^{-1}(u) - F_{\nu_2}^{-1}(u)|^p du \right)^{1/p}$$

Further, the supremum-Wasserstein metric $\bar{d}_p$ is defined between two value distribution functions $Z, Z' \in \mathscr{P}(\mathbb{R})^{\mathcal{S}}$ by

$$\bar{d}_p(Z, Z') = \sup_{s \in \mathcal{S}} d_p(Z(s), Z'(s))$$

A.3.3 Cramér

Definition 7. The Cramér distance ℓ^2 between two distributions $\nu_1, \nu_2 \in \mathscr{P}(\mathbb{R})$, with cumulative distribution functions F_{ν_1}, F_{ν_2} respectively, is defined by:

$$\ell^2(\nu_1, \nu_2) = \left(\int_{\mathbb{R}} (F_{\nu_1}(x) - F_{\nu_2}(x))^2 dx \right)^{1/2}$$

Further, for any pair of value distribution functions $Z, Z' \in \mathscr{P}(\mathbb{R})^{\mathcal{S}}$:

- the supremum-Cramér metric $\bar{\ell}^2$ is defined as

$$\bar{\ell}^2(Z, Z') = \sup_{s \in \mathcal{S}} \ell^2(Z(s), Z'(s)).$$

- Given a vector $\xi \in \mathbb{R}^{\mathcal{S}}$, the ξ-weighted Cramér metric ℓ^2_ξ is defined as

$$\ell^2_\xi(Z, Z') := \sum_{s \in \mathcal{S}} \xi(s) \ell^2(Z(s), Z'(s)).$$

Notation. We abuse notation and denote by $\| \cdot \|^2_\xi$ the ξ-weighted Cramér norm, as we do with the weighted L_2 norm.

A.4 Use of the Arcade Learning Environment

The Arcade Learning Environment (ALE), published in 2013[33], is a software framework for interfacing with emulated Atari 2600 game environments. According to the authors, the idea was to define a new challenging platform for empirically assessing agents designed for general competency. We present its technical description as follows:

"ALE is built on top of Stella[9]*, an open-source Atari 2600 emulator. It allows the user to interface with the Atari 2600 by receiving joystick motions, sending screen and/or RAM information, and emulating the platform. ALE also provides a game-handling layer which transforms each game into a standard reinforcement learning problem by identifying the accumulated score and whether the game has ended. By default, each observation consists of a single game screen (frame): a 2D array of 7-bit pixels, 160 pixels wide by 210 pixels high. The action space consists of the 18 discrete actions defined by the joystick controller, but the game-handling layer also specifies the minimal set of actions needed to play a particular game. When running in real-time, the simulator generates 60 frames per second, and at full speed emulates up to 6000 frames per second. The reward at each time-step is defined on a game by game basis, typically by taking the difference in score or points between frames. An episode begins on the first frame after a reset command is issued, and terminates when the game ends or after a predefined number of frames. The user therefore has access to several dozen games through a single common interface, and adding support for new games is relatively straightforward."*[33]

In our experiments, games are running in real-time (60Hz), just as a human would do. However, directly using Atari 2600 raw coloured frames of 210×160 pixels is usually prohibitive in terms of computation and memory, as it happens in our particular case. For RL agents to deal with that, Dopamine implements by default the preprocessing process unit introduced in [5], whose application has become standard in the community:

"First, to encode a single frame we take the maximum value for each pixel colour value over the frame being encoded and the previous frame. This was necessary to remove flickering that is present in games where some objects appear only in even frames while other objects appear only in odd frames, an artefact caused by the limited number of sprites Atari 2600 can display at once. Second, we then extract the Y channel, also known as luminance, from the RGB frame and rescale it to 84×84*. In practice, this preprocessing is applied to the m most recent frames and stacks them to produce the input to the RL model."*

Hence, considering the typical choice of $m = 4$, the effective states -and thus the network inputs- of our model end up having a size of $84 \times 84 \times 4$. In addition to that, we also use the simple frame-skipping technique presented in [37] (already implemented in Dopamine as well); this makes the agent see and select actions on every k-th frame, being its last action repeated on the skipped ones. In particular, as selecting an action entails much more computation than simply running the emulator one step forward, this technique speeds up the runtime. We also use $k = 4$ in our experiments.

Finally, among all available games through the ALE, we select a small subset of four for the evaluation of our model:

- Seaquest: It is a submarine combat game in which the player controls a water vessel and must avoid, collect, or destroy various objects at different levels of depth.

[9] https://stella-emu.github.io

- Breakout: In the game, a layer of bricks lines the top third of the screen. A ball travels across the screen, bouncing off the top and side walls of the screen. When a brick is hit, the ball bounces away and the brick is destroyed. The player loses a turn when the ball touches the bottom of the screen. To prevent this from happening, the player has a movable paddle to bounce the ball upward, keeping it in play.
- Asterix: The player guides tornado-like object between the stage lines in order to eat hamburgers and avoid the dynamites. The game does not use any buttons and the difficulty increases by increasing the speed of the objects on screen. As the game progresses, the burgers may change into other edible or drinkable objects such as beer kegs, hot dogs, etc.
- Pong: The aim is to defeat an opponent in a simulated table-tennis game by earning a higher score.

It is important to note that, as the scale of scores varies from game to game, during the training process all positive rewards are clipped at 1, all negative rewards at -1, and 0 rewards are left unchanged; this helps us using the same hyper-parameter configuration in all games. Moreover, for games where there is a life counter, the number of lives left is also managed to mark the end of an episode.

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[2] E. Thorndike, *Animal intelligence: Experimental studies*. Routledge, 2017.

[3] W. Schultz, P. Dayan, and P. R. Montague, "A neural substrate of prediction and reward," *Science*, vol. 275, no. 5306, pp. 1593–1599, 1997.

[4] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-dynamic programming*, vol. 5. Athena Scientific Belmont, MA, 1996.

[5] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[6] G. Zheng, F. Zhang, Z. Zheng, Y. Xiang, N. J. Yuan, X. Xie, and Z. Li, "Drn: A deep reinforcement learning framework for news recommendation," in *Proceedings of the 2018 World Wide Web Conference*, pp. 167–176, International World Wide Web Conferences Steering Committee, 2018.

[7] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, "Resource management with deep reinforcement learning," in *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, pp. 50–56, ACM, 2016.

[8] I. Arel, C. Liu, T. Urbanik, and A. G. Kohls, "Reinforcement learning-based multi-agent system for network traffic signal control," *IET Intelligent Transport Systems*, vol. 4, no. 2, pp. 128–135, 2010.

[9] Z. Zhou, X. Li, and R. N. Zare, "Optimizing chemical reactions with deep reinforcement learning," *ACS central science*, vol. 3, no. 12, pp. 1337–1344, 2017.

[10] X. Bu, J. Rao, and C.-Z. Xu, "A reinforcement learning approach to online web systems auto-configuration," in *2009 29th IEEE International Conference on Distributed Computing Systems*, pp. 2–11, IEEE, 2009.

[11] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Thirtieth AAAI conference on artificial intelligence*, 2016.

[12] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu, "Reinforcement learning with unsupervised auxiliary tasks," *arXiv preprint arXiv:1611.05397*, 2016.

[13] B. O'Donoghue, R. Munos, K. Kavukcuoglu, and V. Mnih, "Combining policy gradient and q-learning," *arXiv preprint arXiv:1611.01626*, 2016.

[14] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, pp. 1928–1937, 2016.

[15] M. Fortunato, M. G. Azar, B. Piot, J. Menick, I. Osband, A. Graves, V. Mnih, R. Munos, D. Hassabis, O. Pietquin, *et al.*, "Noisy networks for exploration," *arXiv preprint arXiv:1706.10295*, 2017.

[16] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, p. 354, 2017.

[17] M. G. Bellemare, W. Dabney, and R. Munos, "A distributional perspective on reinforcement learning," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 449–458, JMLR.org, 2017.

[18] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, "Rainbow: Combining improvements in deep reinforcement learning," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[19] G. Barth-Maron, M. W. Hoffman, D. Budden, W. Dabney, D. Horgan, A. Muldal, N. Heess, and T. Lillicrap, "Distributed distributional deterministic policy gradients," *arXiv preprint arXiv:1804.08617*, 2018.

[20] R. Dearden, N. Friedman, and S. Russell, "Bayesian q-learning," in *Aaai/iaai*, pp. 761–768, 1998.

[21] M. G. Azar, R. Munos, and B. Kappen, "On the sample complexity of reinforcement learning with a generative model," *arXiv preprint arXiv:1206.6461*, 2012.

[22] W. Dabney, M. Rowland, M. G. Bellemare, and R. Munos, "Distributional reinforcement learning with quantile regression," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[23] W. Dabney, G. Ostrovski, D. Silver, and R. Munos, "Implicit quantile networks for distributional reinforcement learning," *arXiv preprint arXiv:1806.06923*, 2018.

[24] C. Lyle, M. G. Bellemare, and P. S. Castro, "A comparative analysis of expected and distributional reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 4504–4511, 2019.

[25] M. Rowland, M. G. Bellemare, W. Dabney, R. Munos, and Y. W. Teh, "An analysis of categorical distributional reinforcement learning," *arXiv preprint arXiv:1802.08163*, 2018.

[26] M. G. Bellemare, N. L. Roux, P. S. Castro, and S. Moitra, "Distributional reinforcement learning with linear function approximation," *arXiv preprint arXiv:1902.03149*, 2019.

[27] A. Hallak and S. Mannor, "Consistent on-line off-policy evaluation," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1372–1383, JMLR. org, 2017.

[28] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[29] C. Gelada and M. G. Bellemare, "Off-policy deep reinforcement learning by bootstrapping the covariate shift," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 3647–3655, 2019.

[30] S. P. Meyn and R. L. Tweedie, *Markov chains and stochastic stability*. Springer Science & Business Media, 2012.

[31] J. Tsitsiklis and B. Van Roy, "An analysis of temporal-difference learning with function approximationtechnical," *Report LIDS-P-2322). Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Tech. Rep.*, 1996.

[32] S. C. Jaquette *et al.*, "Markov decision processes with a new optimality criterion: Discrete time," *The Annals of Statistics*, vol. 1, no. 3, pp. 496–505, 1973.

[33] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *Journal of Artificial Intelligence Research*, vol. 47, pp. 253–279, 2013.

[34] P. S. Castro, S. Moitra, C. Gelada, S. Kumar, and M. G. Bellemare, "Dopamine: A research framework for deep reinforcement learning," *arXiv preprint arXiv:1812.06110*, 2018.

[35] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

[36] H. V. Hasselt, "Double q-learning," in *Advances in Neural Information Processing Systems*, pp. 2613–2621, 2010.

[37] M. G. Bellemare, J. Veness, and M. Bowling, "Investigating contingency awareness using atari 2600 games," in *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.